21世纪高等学校计算机教育实用规划教材

移动Web开发技术

袁伟华 主编
潘志宏 黄耿生 张译匀 编著

清华大学出版社

21 世纪高等学校计算机教育实用规划教材

移动 Web 开发技术

袁伟华　主编
潘志宏　黄耿生　张译匀　编著

清华大学出版社
北　京

内 容 简 介

本书全面介绍了 HTML5 中的标签、HTML5 数据存储，定位和离线应用，CSS3 的基础知识以及 jQuery Mobile、sencha-touch 和 PhoneGap 三个框架，重点介绍了 HTML5 和 CSS3 在开始移动 Web 中的应用，并通过相应的例子来阐述如何使用相关的知识来开发移动 Web 程序。

全书共分 8 章：第 1 章为移动 Web 开发技术概论，主要介绍移动 Web 的发展历程，包括移动 Web 和桌面 Web 的区别；第 2～4 章为 HTML5 的知识内容，着重讨论 HTML5 的标签和数据存储、定位和离线应用；第 5 章为 CSS3 的知识，重点介绍 CSS3 在移动 Web 开发中的应用；第 6～8 章为 jQuery Mobile、sencha-touch 和 PhoneGap 框架的知识内容，重点介绍如何利用它们开发跨平台的移动 Web 程序。全书提供了大量应用实例，可以让读者快速掌握开发要点。

本书适合作为高等职业技术院校，普通高等院校计算机、计算机专业、电子商务专业高年级专科生、本科生的教材，同时可供对移动 Web 开发感兴趣的开发人员、广大科技工作者和研究人员参考。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

图书在版编目(CIP)数据

移动 Web 开发技术/袁伟华主编.--北京：清华大学出版社，2016
21 世纪高等学校计算机教育实用规划教材
ISBN 978-7-302-43463-4

Ⅰ.①移… Ⅱ.①袁… Ⅲ.①移动终端－应用程序－程序设计－高等学校－教材
Ⅳ.①TN929.53

中国版本图书馆 CIP 数据核字(2016)第 078261 号

责任编辑：刘向威 薛 阳
封面设计：
责任校对：徐俊伟
责任印制：

出版发行：清华大学出版社
网 址：http://www.tup.com.cn，http://www.wqbook.com
地 址：北京清华大学学研大厦 A 座 **邮 编**：100084
社 总 机：010-62770175 **邮 购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn,010-62795954
印 刷 者：
装 订 者：
经 销：全国新华书店
开 本：185mm×260mm **印 张**：12.5 **字 数**：312 千字
版 次：2016 年 6 月第 1 版 **印 次**：2016 年 6 月第1 次印刷
印 数：1～ 000
定 价： .00 元

产品编号：066830-01

出 版 说 明

随着我国高等教育规模的扩大以及产业结构调整的进一步完善，社会对高层次应用型人才的需求将更加迫切。各地高校紧密结合地方经济建设发展需要，科学运用市场调节机制，合理调整和配置教育资源，在改革和改造传统学科专业的基础上，加强工程型和应用型学科专业建设，积极设置主要面向地方支柱产业、高新技术产业、服务业的工程型和应用型学科专业，积极为地方经济建设输送各类应用型人才。各高校加大了使用信息科学等现代科学技术提升、改造传统学科专业的力度，从而实现传统学科专业向工程型和应用型学科专业的发展与转变。在发挥传统学科专业师资力量强、办学经验丰富、教学资源充裕等优势的同时，不断更新教学内容、改革课程体系，使工程型和应用型学科专业教育与经济建设相适应。计算机课程教学在从传统学科向工程型和应用型学科转变中起着至关重要的作用，工程型和应用型学科专业中的计算机课程设置、内容体系和教学手段及方法等也具有不同于传统学科的鲜明特点。

为了配合高校工程型和应用型学科专业的建设和发展，急需出版一批内容新、体系新、方法新、手段新的高水平计算机课程教材。目前，工程型和应用型学科专业计算机课程教材的建设工作仍滞后于教学改革的实践，如现有的计算机教材中有不少内容陈旧（依然用传统专业计算机教材代替工程型和应用型学科专业教材），重理论、轻实践，不能满足新的教学计划、课程设置的需要；一些课程的教材可供选择的品种太少；一些基础课的教材虽然品种较多，但低水平重复严重；有些教材内容庞杂，书越编越厚；专业课教材、教学辅助教材及教学参考书短缺，等等，都不利于学生能力的提高和素质的培养。为此，在教育部相关教学指导委员会专家的指导和建议下，清华大学出版社组织出版本系列教材，以满足工程型和应用型学科专业计算机课程教学的需要。本系列教材在规划过程中体现了如下一些基本原则和特点。

（1）面向工程型与应用型学科专业，强调计算机在各专业中的应用。教材内容坚持基本理论适度，反映基本理论和原理的综合应用，强调实践和应用环节。

（2）反映教学需要，促进教学发展。教材规划以新的工程型和应用型专业目录为依据。教材要适应多样化的教学需要，正确把握教学内容和课程体系的改革方向，在选择教材内容和编写体系时注意体现素质教育、创新能力与实践能力的培养，为学生知识、能力、素质协调发展创造条件。

（3）实施精品战略，突出重点，保证质量。规划教材建设仍然把重点放在公共基础课和专业基础课的教材建设上；特别注意选择并安排一部分原来基础比较好的优秀教材或讲义修订再版，逐步形成精品教材；提倡并鼓励编写体现工程型和应用型专业教学内容和课程体系改革成果的教材。

(4) 主张一纲多本,合理配套。基础课和专业基础课教材要配套,同一门课程可以有多本具有不同内容特点的教材。处理好教材统一性与多样化,基本教材与辅助教材,教学参考书,文字教材与软件教材的关系,实现教材系列资源配套。

(5) 依靠专家,择优选用。在制订教材规划时要依靠各课程专家在调查研究本课程教材建设现状的基础上提出规划选题。在落实主编人选时,要引入竞争机制,通过申报、评审确定主编。书稿完成后要认真实行审稿程序,确保出书质量。

繁荣教材出版事业,提高教材质量的关键是教师。建立一支高水平的以老带新的教材编写队伍才能保证教材的编写质量和建设力度,希望有志于教材建设的教师能够加入到我们的编写队伍中来。

21 世纪高等学校计算机教育实用规划教材编委会
联系人:魏江江 weijj@tup.tsinghua.edu.cn

前　言

最近几年，随着宽带无线移动通信技术的发展，移动互联网得到了迅速的发展，继计算机、互联网之后，移动互联网正掀起第三次信息技术革命的浪潮，IT 业界的新技术、新应用不断涌现，Web 应用技术的不断创新，并且已经渗透到社会生活的方方面面，从而孕育了巨大的市场潜力和发展机会。

与此同时，HTML5 和 CSS3 技术进一步结合，HTML5 首先强化了 Web 网页的表现能力，其次追加了本地数据库等相关功能。所谓的 HTML5 实际上是指包括 HTML、CSS 和 JavaScript 在内的一套技术组合，目前支持 HTML5 的主流浏览器有 Google Chrome、Firefox、IE 9 以上版本的浏览器。HTML5 新增的视频、音频、画布、离线应用等功能为移动 Web 开发带来了便利。新技术不仅能很好地适应移动终端界面，而且很大程度上减少了代码冗余，提高了移动设备运行效率。随着 HTML5 和 CSS3 的发展，基于 HTML5 的应用在未来更有可能对移动互联网领域产生巨大的影响。

正是在此时代背景之下，我们就很有必要去学习新技术和新知识，以适应时代发展的要求，本书重点介绍 HTML5 和 CSS3 的新特性，以及如何利用 jQuery Mobile、sencha-touch 和 PhoneGap 三个移动 Web 开发框架去开发移动 Web 项目。

本书的适合人群主要有：具有一定技术基础的 Web 前端开发工程师；具有一定美工基础的 UI 设计师；开设了 Web 开发等相关专业的高等职业技术院校和相关的培训机构的师生。此外，也适合熟悉 Java、PHP、ASP. NET 等后端 Web 技术的开发者阅读。

本书是配合广东行政职业学院电子商务专业重点建设的技术教材，在编写过程中，多位老师付出了辛勤的劳动，并提出了许多中肯的建议，在此，对参与编写的各位老师表示诚挚的感谢。

另外，由于时间和水平有限，在本书的编写过程中可能存在一些对 HTML5 等技术认识不全面或者表述疏漏的地方，敬请读者批评和指正，我们谨以最真诚的心希望与读者共同交流，共同成长。

编　者

2016 年 3 月

目　　录

]]

第1章 移动 Web 开发技术概述

本章学习目标

- 智能机的发展过程
- 智能手机的 Web 浏览器
- HTML5 移动 Web 应用

移动互联网在当前时代得到了快速的发展,Android 与 iOS 等新技术在移动互联网领域成为市场的主流,市场潜力巨大,同时跨平台的 HTML5 应用在未来更有可能对移动互联网领域起到巨大的影响。

本书将主要介绍 HTML5 的一些新标准及新特性,同时结合移动互联网领域,将为读者带来全新的技术体验,甚至可以让只有 Web 技术基础的读者同样能参与移动互联网开发。在本书中,将学习如何构建适应性强、响应迅速并且符合标准的移动 Web 站点,并确保其可以在任意移动浏览器上运行。一些简单的开发提示和技巧将改进小尺寸屏幕中的 Web 可用性,并且可以进一步充实移动 Web 站点,使其适用于高级智能手机浏览器(具有电子邮件、桌面功能的 Web 浏览等集成 Internet 功能的高端手机中的浏览器),能够呈现完整的 HTML 并实现专有扩展。开发工作完成后,将了解如何在实际移动设备上进行全方位的测试,优化移动 Web 页面以便于网络传输。

本章首先介绍智能机的发展过程,再介绍智能手机的 Web 浏览器,最后介绍智能手机的 Web 浏览器以及移动 Web 和桌面 Web 的区别。

1.1 智能机的发展

Android 和 iOS 平台的智能手机伴随着移动互联网的发展,让越来越多的应用程序在其平台下的软件市场发布软件。同时,各家公司为了使自己的产品线能够更快地在移动互联网上占有市场份额,也纷纷将自己的产品线布局到移动设备上。因此,移动互联网大战一触即发。

1. WAP 1.0 时代

实际上,早在 2000 年的时候,移动互联网已经进入了人们的生活,这个时候手机所提供的功能有限,基本上都是只提供铃声、彩铃、图片等服务内容。这种服务使相当一部分创业者在短期内得到可观的收入,这个时代通常被称为 SP 时代。但是,这种服务只能满足部分手机用户的低层次需求。

2. WAP 2.0 时代

直到 2006 年，智能手机得到不断发展，手机用户的需求开始产生变化，各种新的手机应用不断推出，如新闻类资讯、即时聊天等。事实上，这些新的应用也只不过是在 SP 时代功能基础上的升级，这就是 WAP 2.0 时代。

3. 3G 时代

进入 3G 时代，移动互联网发展速度非常快，特别是以谷歌、苹果为首的 Android 和 iOS 平台的手机推出后，智能手机的功能逐渐变得非常强大，例如，WiFi 无线联网、蓝牙、加速计、指南针、重力感应、数据存储等功能，让智能手机变得不再是一部简单的手机。

Android 平台手机和 iOS 平台的 iPhone 在中国乃至全球的手机市场份额不断扩大，越来越多的用户愿意尝试使用这种新平台的智能手机，其原因有以下几点。

1）硬件设备的提升

手机经过十多年的发展，其硬件设备相比十年前已经发生翻天覆地的变化。各种单核、双核，甚至四核 CPU 的智能手机不断推出，其运算速度得到很大的提升，为大型软件和游戏提供了最好的硬件基础。

2）平台的开放性

Android 平台以免费开源的方式打破了过去手机操作系统的封闭性，让各个手机制造商可以利用 Android 平台制造出用户体验更好、功能更强大的手机。虽然 iOS 平台没有像 Android 那样开放源代码，但是 iOS 和 Android 都提供非常丰富的 API 接口和文档，开发者可以通过其提供的 API 接口开发出极具创意的应用程序。

3）更好的用户体验

过去，Symbian 系统占据着整个手机系统市场的半壁江山。然而使用 Symbian 系统的手机只是一款符合手机用户操作习惯的移动电话。但是运行 Android 或 iOS 的手机更像是一款移动掌上设备。它们不仅提供手机最基本的功能，还能使用许多丰富的软件、游戏开发接口以及可定制的用户界面库。这就使得手机用户可以使用用户体验更好、更具创意的应用软件。

4）丰富的应用程序

目前基于 iOS 平台的 App Store 软件商店上软件数量已经超过 30 万，Android 平台的 Android Market 软件市场上软件数量更是已经超过 App Store。以目前这样的应用程序数量发展情况来看，手机用户没有不使用它们的理由。

5）创业机会

Android 平台的开发采用的是 Java 语言，它是目前最流行的语言之一，而 iOS 平台则采用类似 C 的 Object-C 语言。这两种语言对于开发者来说并不陌生，要真正去学习这两个平台的开发，其成本非常低。基于此原因，很多开发者都会利用这个契机去实现创业梦想，同时通过开发各种应用程序，并结合移动网络、社交功能、网上支付等重要功能，不断寻找这些应用程序的赢利方向。

1.2 智能手机的 Web 浏览器

随着智能手机的发展，Android 平台手机、iOS 平台的 iPhone、黑莓(BlackBerry)手机不断推出各种应用程序。然而，它们都各自内置有一款令人感到陌生的应用程序，就是移动

Web 浏览器，例如：

(1) Android 中的 Android Browser；

(2) iOS 中的 Mobile Safari；

(3) BlackBerry 中的 Webkit 浏览器；

(4) Symbian S60 中的 Web Browser for S60。

这些移动 Web 浏览器不同于过去的 WAP 浏览器，它能识别和解释 HTML、CSS、JavaScript 等代码，而且它们都有一个共同的特点就是其浏览器的核心都是基于 Webkit。随着 iOS 5.0 版本的发布，Safari 浏览器已经成为移动端表现最好的 Web 浏览器。

虽然 Symbian 最新版本已经开始自带有 Webkit 核心的浏览器，但从目前情况来看，它并没有像其余三种平台那样得到广泛使用。

Webkit 实际上是一种浏览器引擎，同时也是一个开源的项目，其起源可以追溯到 Kool Desktop Environment(KDE)。在桌面浏览器中，Chrome 谷歌浏览器、Apple 的 Safari 浏览器都已经内置了 Webkit 引擎，并支持 HTML5 和 CSS3 特性。在移动端方面，黑莓更是直接将 Webkit 浏览器内置到平台当中。

Mobile Safari 和 Android Browser 作为两大平台内置的移动 Web 浏览器，更是继承各自桌面端浏览器的特点，既支持 Webkit 引擎特性，也支持 HTML5 和 CSS3 的多项特性。

移动 Web 浏览器所带来真正意义上的改变，就是可以通过浏览器直接访问任何通过 HTML 静态语言或类似 PHP、ASP.NET 等动态语言构建的 Web 网站或应用程序，而不仅仅是 WAP 网站。

智能手机的移动 Web 浏览器具有以下几个特点。

(1) 有限的屏幕尺寸。

由于智能手机屏幕尺寸的原因，例如，iPhone 4 的实际屏幕尺寸是 320×480 或 480×320(横向)，传统的 Web 网站在移动 Web 浏览器中很难给用户完美体验，因此需要根据智能手机的屏幕大小定制移动版 Web 应用程序。

(2) 触屏、缩放。

在移动互联网时代，触屏手机的大规模应用及手机应用范围的转变，使触摸屏成为行业的大趋势。其中，Web 页面浏览、下载、消费等都成为主要应用场景，用户可以直接在屏幕上进行触摸、点击来完成交互操作。

(3) 硬件设备的提升。

智能手机硬件的不断升级换代，使 Apple 的 Mobile Safari 和 Android 的 Android Browser 两种移动 Web 浏览器得到更好的发展，同时能够充分利用 CPU 等硬件的更高性能去做更多的事情。

(4) 基于 Webkit 内核。

移动 Web 浏览器支持各种 Web 技术标准，并且支持 HTML5 和 CSS3 的大部分标准。

1.3 HTML5 的移动 Web 应用

基于 Webkit 内核的浏览器的一个最大特点就是支持 HTML5 和 CSS3 标准。基于 HTML5、CSS3 和 JavaScript 的移动应用程序将会是未来的趋势。HTML5 标准定义的规

范非常广泛,以下标准在目前的移动浏览器中已得到支持。

1. Canvas 绘图

HTML5 标准最大的变化就是支持 Web 绘图功能。Canvas 绘图功能非常强大,如图形绘制、路径绘制、变形、像素绘图、动画等。用户可以通过获取 XML 中 DOM 元素 Canvas,并调用其渲染上下文的 Context 对象,使用 JavaScript 进行图形绘制。

2. 多媒体

Apple 的 iOS 在平台默认情况下不支持播放 Flash 文件。HTML5 的多媒体标准就是 Apple 公司的最佳解决方案,因为它不需要任何插件,只需要几个页面标签就能实现多媒体的播放。

HTML5 标准中的多媒体,Video 视频和 Audio 音频正好弥补了多年来需要插件才能播放 Flash 模式的缺陷。现在只需要利用 Video 和 Audio 通过简单几行页面代码,就能播放互联网上的各种视频文件。可是,各家浏览器提供商对多媒体标准所支持的播放格式不一致,导致多媒体标准的发展无法像其他标准那样大放异彩。例如,Google 的 Chrome 最新版本支持的多媒体视频格式是:ogg、MPEG4、WebM,而 Apple 的 Safari 则只支持 MPEG4。因此,真正在移动设备的 Web 浏览器上实现多媒体功能还尚需时日。

3. 本地存储

为了满足本地存储数据的需求,HTML5 标准中新增两种存储机制,即 Web Storage 和 Web SQL Database。前者通过提供 key/value 方式存储数据,后者通过类似关系数据库的形式存储数据。移动 Web 浏览器对 Web Storage 的支持情况比较理想。

4. 离线应用

HTML5 标准规范提供一种离线应用的功能。当支持离线应用的浏览器检测到清单文件(Manifest File)中的任何资源文件时,便会下载对应的资源文件,将它们缓存到本地,同时它也保证本地资源文件的版本和服务器上的保持一致。

对于移动设备来说,当无网络状态可用时,Web 浏览器便会自动切换到离线状态,并读取本地资源以保证 Web 应用程序继续可用。

5. 使用地理位置

现在,很多现代浏览器中都实现了一个神奇的功能,它能实时获取到当前在地图上所在的位置。虽然地理定位标准严格上来说并不属于 HTML5 标准规范的一部分,但它已经逐渐得到大部分浏览器的支持。

6. 移动 Web 框架

因为有了 Webkit 和 HTML5 的支持,越来越多的 Web 开发者开始研究基于移动平台的 Web 应用框架,例如,基于 jQuery 页面驱动的 jQuery Mobile.,基于 ExtJS 架构的 Sencha Touch,以及能打通 Web 和 Native 两者之间通道的 PhoneGap 框架。

目前基于 HTML5 移动 Web 框架存在两种不同的开发模式:基于传统 Web 的开发和基于组件式的 Web 开发。

基于传统 Web 的开发模式,就是在传统 Web 网站上,根据移动设备(如手机)平台的特点展示其移动版的 Web 站点。目前最能体现该开发模式优势的 Web 框架是 jQuery Mobile。通过使用 CSS3 的新特性,Media Queries 模块在实现一个站点的同时能自适应任何设备,包括桌面计算机和智能手机。

基于组件式的 Web 开发有些类似于 Ext 所提供的富客户端开发模式，在该模式下几乎所有的组件或视图都封装在 JavaScript 内，然后通过调用这些组件展示 Web 应用。这种模式的最佳代表是 Sencha Touch。

1.4 移动 Web 和桌面 Web

从根本上说，只有一种 Web，Web 内容是可以使用各种 Web 浏览器查看的标准化标记、样式、脚本和多媒体。在本书中，按照惯例将这种 Web 称为桌面 Web。我们可以在台式计算机、笔记本、上网本上通过 Firefox、Opera 或 Internet Explorer 等浏览器访问桌面 Web，进行网上冲浪。

桌面 Web 由通过 TCP/IP 计算机网络连接在一起的大量服务器构成。这种服务器称为 Web 服务器，很多 Web 服务器实现超文本传输协议（HTTP）共享文档和文件。Web 服务器通过统一资源标识符（Uniform Resource Identifier，URI）提供对文本文件、标记文档和二进制资源的访问。

在 HTTP 请求中，客户端向 Web 服务器发送所需资源的 URI 以及一组请求头，其中一个请求头包含 MIME 类型列表，该列表公布客户端支持的内容类型。

在 HTTP 响应中，Web 服务器除了向客户端发送请求的文档（标记、文本或二进制文件）外，还会附带另外一组头，其中一个头包含 MIME 类型，描述传输到客户端的文档的文件类型。

移动 Web 在桌面 Web 的基础上添加了新的 MIME 类型、标记语言、文档格式和最佳实践，为小尺寸屏幕提供优化的 Web 内容，并可解决移动设备上的资源限制、Web 浏览器可用性差等问题。

移动 Web 在 Web 生态系统中引入了一些新的组件，包括：针对移动设备进行了优化的标记语言和样式；可区分移动标记和桌面 HTML 的 MIME 类型；具有大量功能的浏览器客户端；使内容更适合上述客户端的网络代理。

移动 Web 开发是一门全新的学科，原因如下。

(1) 移动 Web 生态系统是全新的。移动 Web 使用桌面 Web 的既有知识，但它也有一些从移动设备独有的特性衍生出来的新的最佳实践和疑难问题。桌面并不适用。带宽占用量是一个比较关键的问题，即使对智能手机也是如此。使用 JavaScript 架构以及异步 JavaScript 和 XML（AJAX）等 Rich Web 2.0 功能时务必要谨慎，否则有电池电量耗尽的风险。运营商经常会控制和阻止移动 Web 站点的流量。在移动标记传输到移动浏览器的途中，代码转换代理经常会尝试重置移动标记的格式。最后，必须编写一些保护性的程序，降低代码转换器暴露以及出现移动网络问题的概率。

(2) 移动 Web 用户是全新的。移动 Web 用户采用独特的使用模式和导航方法。移动用户具有强烈的目标导向性和位置感知能力。在影响移动用户的移动 Web 浏览体验的主要因素中，即包括在服务区内外漫游时的网络访问问题。实际上，对成本敏感的移动用户宁可取消网络交易，也不愿意因错误操作而支付费用。

(3) 移动 Web 浏览器是全新的。移动浏览器具有其他浏览器所不具备的优势，同时也有一些与众不同的问题及相应的解决方法。Web 标准实现不彻底是经常出现的问题。

Web 页面格式错误会对移动设备产生严重的影响,包括浏览器崩溃或设备重置。用户非常需要 JavaScript 和 AJAX 等高级 Web 功能,但这些功能会影响电池的使用寿命。市场上有数十家移动浏览器供应商,确保 Web 标准遵从性的重担就落在了原始设备制造商(Original Equipment Manufacturer,OEM)和运营商肩上。

小 结

在本章中,首先介绍了智能机的发展过程,以及相应的浏览器,然后介绍了 HTML5 的移动 Web 应用,最后介绍了移动 Web 和桌面 Web 的区别,通过本章的学习,可以对移动 Web 开发技术有个初步的了解。

第2章 HTML5与移动Web

本章学习目标

- HTML5 标签元素
- 移动 Web 页面布局
- 移动 Web 实例

标准通用标记语言下的一个应用 HTML 标准自 1999 年 12 月发布的 HTML4.01 后，后继的 HTML5 和其他标准被束之高阁，为了推动 Web 标准化运动的发展，一些公司联合起来，成立了一个叫做 WHATWG（Web Hypertext Application Technology Working Group，Web 超文本应用技术工作组）的组织。WHATWG 致力于 Web 表单和应用程序，而 W3C（World Wide Web Consortium，万维网联盟）专注于 XHTML2.0。在 2006 年，双方决定进行合作，来创建一个新版本的 HTML。

HTML5 草案的前身名为 Web Applications 1.0，于 2004 年被 WHATWG 提出，于 2007 年被 W3C 接纳，并成立了新的 HTML 工作团队。HTML5 的第一份正式草案已于 2008 年 1 月 22 日公布。HTML5 仍处于完善之中。然而，大部分现代浏览器已经具备了某些 HTML5 支持。

2012 年 12 月 17 日，万维网联盟（W3C）正式宣布凝结了大量网络工作者心血的 HTML5 规范已经正式定稿。根据 W3C 的发言稿称："HTML5 是开放的 Web 网络平台的奠基石。"

2013 年 5 月 6 日，HTML5.1 正式草案公布。该规范定义了第 5 次重大版本，第一次要修订万维网的核心语言：超文本标记语言（HTML）。在这个版本中，新功能不断推出，以帮助 Web 应用程序的作者，努力提高新元素互操作性。本次草案的发布，从 2012 年 12 月 27 日至今，进行了多达近百项的修改，包括 HTML 和 XHTML 的标签，相关的 API、Canvas 等，同时 HTML5 的图像 img 标签及 svg 也进行了改进，性能得到进一步提升。

2014 年 10 月 29 日，万维网联盟经过近八年的艰辛努力，HTML5 标准规范终于最终制定完成了，并已公开发布。

HTML5 手机应用的最大优势就是可以在网页上直接调试和修改。原先应用的开发人员可能需要花费非常大的力气才能达到 HTML5 的效果，不断地重复编码、调试和运行，这是首先得以解决的一个问题。因此也有许多手机杂志客户端是基于 HTML5 标准，开发人员可以轻松调试修改。HTML5 将会取代 1999 年制定的 HTML4.01、XHTML1.0 标准，以期能在互联网应用迅速发展的时候，使网络标准达到符合当代的网络需求，为桌面和移动平台带来无缝衔接的丰富内容。

支持 HTML5 的浏览器包括 Firefox(火狐浏览器),IE 9 及其更高版本,Chrome(谷歌浏览器),Safari,Opera 等;国内的遨游浏览器(Maxthon),以及基于 IE 或 Chromium(Chrome 的工程版或称实验版)所推出的 360 浏览器、搜狗浏览器、QQ 浏览器、猎豹浏览器等国产浏览器同样具备支持 HTML5 的能力。

HTML5 提供了一些新的元素和属性,例如,<nav>(网站导航块)和<footer>。这种标签将有利于搜索引擎的索引整理,同时更好地帮助小屏幕装置和视障人士使用。除此之外,还为其他浏览要素提供了新的功能,如<audio>和<video>标记。一些过时的 HTML4 标记将被取消。其中包括纯粹显示效果的标记,如<font>和<center>,它们已经被 CSS 取代。

在本章中,主要结合移动设备的特点,介绍了 HTML5 中新增的语义化标签元素,以及在移动 Web 浏览器下 Web 页面布局的知识及例子。HTML5 标准添加的新元素当中,用于标识常见页面结构的包括:section、header、footer、nav、article 和 mark 等。

2.1 HTML5 的优势

(1) 让 Web 再次回归到富客户端地步,减少了对第三方插件的依赖。

在之前的 HTML4 的标准中并没有对于视频、音频以及其他的富客户端技术支持得非常好,这就使得 Flash 和 SilverLight 变得异常的成功。而在 HTML5 新标准中原生的就支持音频、视频、画布等技术。让 Web 程序拥有更多富客户端表现的方式,而且让 Web 程序更加独立,更好地适应多种形式的客户端。

(2) 对本地离线存储的更好的支持。

由于之前想在客户端保存一些数据都是由 cookie 完成的,但是 cookie 不适合大量数据的存储,因为它们由每个对服务器的请求来传递,这使得 cookie 速度很慢而且效率也不高。HTML5 提供了两种在客户端存储数据的新方法:localStorage(没有时间限制的数据存储)和 sessionStorage(针对一个 session 的数据存储)。

在 HTML5 中,数据不是由每个服务器请求传递的,而是只有在请求时使用数据。它使在不影响网站性能的情况下存储大量数据成为可能。对于不同的网站,数据存储于不同的区域,并且一个网站只能访问其自身的数据。HTML5 使用 JavaScript 来存储和访问数据。有了本地数据库的支持,让一些简单的离线应用也成为可能。

(3) 新的特殊内容元素,更好地支持 SEO。

现在所有的站点基本上都是 DIV+CSS 布局,几乎所有的文章标题、内容、辅助介绍等都用 DIV 容器来承载。搜索引擎在抓取页面内容时,因为没有明确的容器的含义只能去猜测这些标签容器承载的是文章标题还是文章内容等,HTML5 新标准中直接添加了拥有具体含义的 HTML 标签,如 article、footer、header、nav、section。

(4) 更加智能的表单标签。

之前的表单标签,仅仅是简单的类型的约束,比如文本框、文本域、下拉列表等,而跟业务结合紧密的表单标签数据校验等控制都没有很好的支持,而使用这些技术基本上都是跟第三方的 JavaScript 控件进行结合使用,但是这些第三方总会涉及版本控制、浏览器兼容性、非标准等一系列的问题,而在 HTML5 的标准中直接添加了智能表单,让这一切都变得

那么的简单，比如 calendar、date、time、email、url、search。

(5) HTML5 引入了画布。

画布的引入使得 Web 端生成动画效果、制作 Web 游戏、更好的交互体验设计都增加了无限的变数。HTML5 的 canvas 元素使用 JavaScript 在网页上绘制图像。画布是一个矩形区域，可以控制其每一像素。canvas 拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。

(6) 支持多线程。

在不影响 UI 更新及浏览器与用户交互的情况下，前端做大规模运算，只能通过 setTimeout 等去模拟多线程。而新的标准中，JavaScript 新增的 HTML5 Web Worker 对象原生的就支持多线程。

(7) WebSocket 让跨域请求、长连接、数据推送等一切都变得那么简单。

WebSocket 是在一个(TCP)接口进行双向通信的技术，它是 HTML5 规范新引入的功能，用于解决浏览器与后台服务器双向通信的问题，使用 WebSocket 技术，后台可以随时向前端推送消息，以保证前后台状态统一，在传统的无状态 HTTP 中，这是无法做到的。

(8) 更好的异常处理。

HTML5 浏览器将在错误语法的处理上更加灵活，HTML5 在设计时保证旧的浏览器能够安全地忽略掉新的 HTML5 代码。与 HTML4 相比，HTML5 给出了解析的完整规则，让不同的浏览器即使在发生语法错误时也能返回完全相同的结果。

(9) 文件 API 让文件上传和操纵文件变得更简单。

由于项目中经常遇到用 Web 应用控制操作本地文件，而之前都是使用一些富客户端技术比如 Flash、ActiveX、Silverlight 等，面对文件，JavaScript 无能为力。而在 HTML5 中新提供的 FHTML5 File API 让 JavaScript 可以轻松上阵了。

(10) 编辑、拖放、微数据、浏览历史管理、地理信息接口 API、设备硬件操作 API 等新功能。

当然 HTML5 不是孤立的，JavaScript API 的增强，让 JavaScript 变成异常强大的未来的编程武器。CSS3 给未来 Web 应用也带来了极大的新的挑战。由于 HTML5 标准化的支持，相信未来 Web 技术真正地可以跑在任何客户端，也让 Web 应用更加独立，更加轻松地融入到各个客户端。

通过上面 HTML5 的新特点，容易了解到 HTML5 的新特性带给开发者的是更友好更丰富的本地处理 API，更智能更优雅的 HTML 标签，更强的本地处理功能，通信也进一步加强。作为开发者，当 Adobe 公司宣布放弃 Flash，把最大的精力放到 HTML5 的开发上的时候，就可能会看到这些趋势，当微软选择了 HTML5 而放弃了 Silverlight 继续升级的时候，那就基本上也没有什么好选择的了。HTML5 的发力，的确给每个开发者都带来了机会。

2.2 HTML4 与 HTML5 的区别

HTML5 是 HTML 标准的下一个版本。越来越多的程序员开始用 HTML5 来构建网站。如果同时使用 HTML4 和 HTML5，会发现用 HTML5 从头构建，比从 HTML4 迁移到 HTML5 方便很多。虽然 HTML5 没有完全颠覆 HTML4，它们还是有很多相似之处，

但是它们也有一些关键的不同。本文就列出了它们之间 10 个关键的不同之处。

(1) HTML5 标准还在制定中。

首先要注意的是,HTML5 虽然现在很流行,但是 HTML5 标准还在制定中,标准仍在改变。HTML4 已经十多年了,不会有任何改变了。

(2) 简化的语法。

HTML5 简化了很多细微的语法,例如 doctype 的声明,用户只需要写<!doctype html>就行了。HTML5 兼容 HTML4 和 HTML1.0,但是与 SGML 不兼容。

(3) <canvas>标签替代 Flash。

Flash 给很多 Web 开发者带来了麻烦,要在网页上播放 Flash 需要一堆代码和插件。<canvas>标签使得开发者只要使用一个标签就能和用户产生 UI 交互。虽然目前<canvas>标签还不能实现 Flash 的所有功能,但是很快<canvas>就会让 Flash 看起来过时了。

(4) 新增<header>和<footer>标签。

HTML5 设计的一个原则是更好地体现网站的语义性,所以增加了<header>和<footer>这样的标签,用来明确表示网页的结构。

(5) 新增<section>和<article>标签。

与<header>和<footer>类似,<section>和<article>也有利于清晰化网页的结构,更有利于 SEO。

(6) 新增<menu>和<figure>标签。

<menu>可以被用于创建传统的菜单,也可以用于工具栏和上下文菜单。<figure>标签使得网页文字和图片的排版更专业。

(7) 新增<audio>和<video>标签。

这两个标签可能是 HTML5 里面最有用的两个标签。顾名思义,这两个标签是用来播放音频和视频的。

(8) 全新的表单。

HTML5 对<form>和<forminput>标签进行了大量修改,添加了很多新的属性,也修改了很多属性。

(9) 删除<b>和<font>标签。

这个改进很多人还无法理解。他们不认为删除这两个标签对代码的改进有很大的帮助。官方的解释是应该用 CSS 来替代这两个标签。但是对于简单的文本,这两个标签还是很方便的。

(10) 删除<frame>、<center>和<big>标签。

以上 10 点只是 HTML5 和 HTML4 差别的很小一部分,当然还有很多其他的变化,这里不一一介绍。

2.3 HTML5 的新标签

在之前的 HTML 页面中,基本上都是用 DIV+CSS 的布局方式。而搜索引擎去抓取页面内容的时候,它只能猜测某个 DIV 内的内容是文章内容容器,或者是导航模块的容器,

或者是作者介绍的容器等。也就是说整个 HTML 文档结构定义不清晰，HTML5 中为了解决这个问题，专门添加了：页眉、页脚、导航、文章内容等跟结构相关的结构元素标签。

在介绍这些新标签之前，先看一个普通的页面的布局方式，如图 2.1 所示。

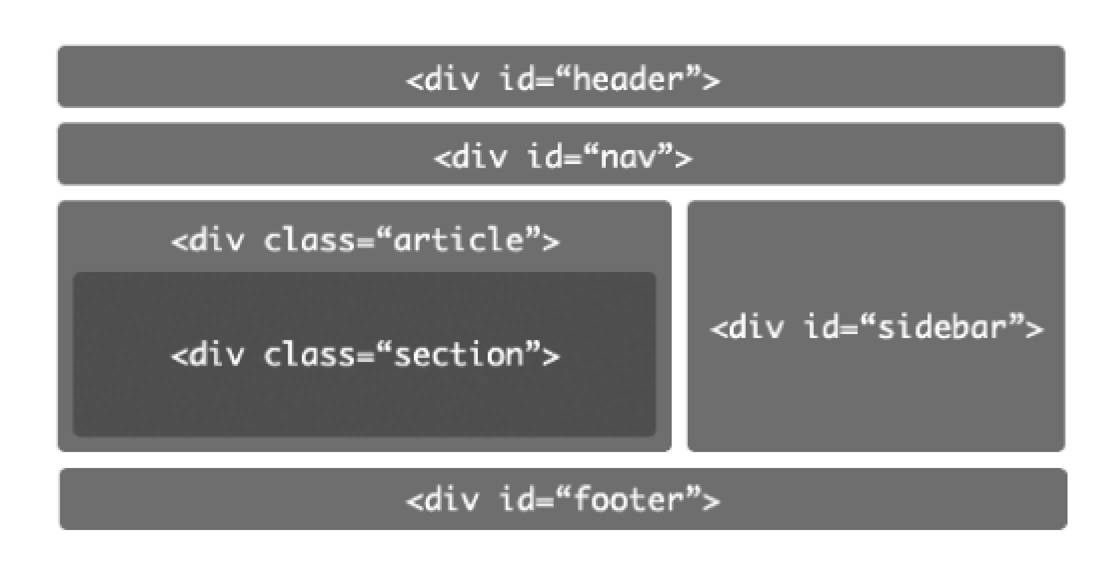


图 2.1　普通的页面布局

从图 2.1 中可以非常清晰地看到一个普通的页面，会有头部、导航、文章内容，还有附着的右边栏，以及底部等模块，通过 class 进行区分，并通过不同的 CSS 样式来处理。而 HTML5 新标签带来的新的布局则如图 2.2 所示。

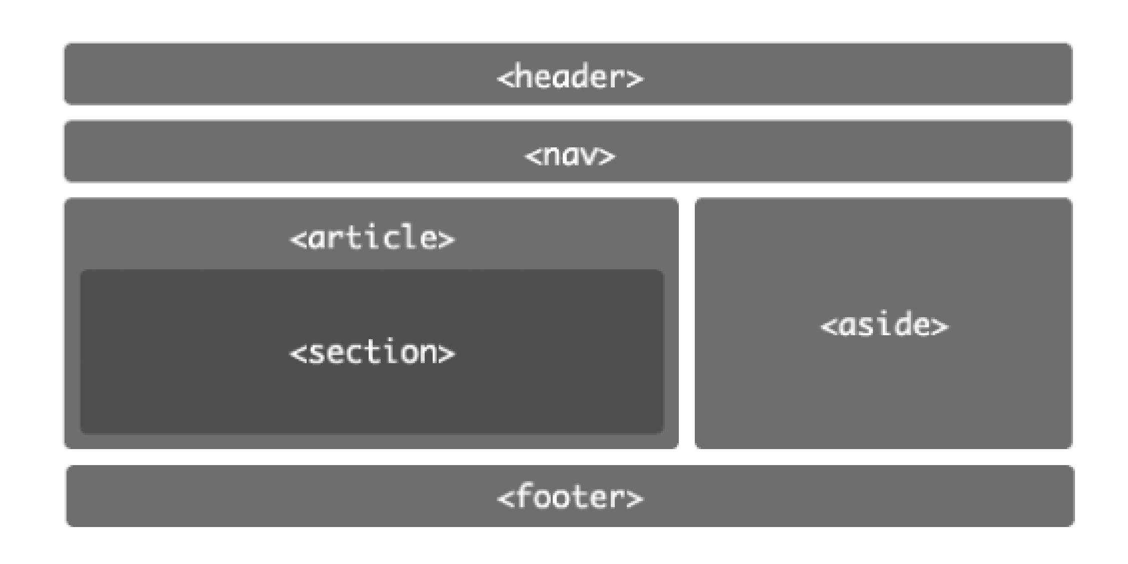


图 2.2　HTML5 新标签布局

图 2.2 相关的 HTML 代码如下。

```
<body>
  <header>...</header>
  <nav>...</nav>
  <article>
    <section>
      ...
    </section>
  </article>
  <aside>...</aside>
  <footer>...</footer>
</body>
```

有了上面的直接的感官的认识后，下面来介绍 HTML5 中的相关结构标签。

2.3.1　section 标签

<section>标签，定义文档中的节。比如章节、页眉、页脚或文档中的其他部分。一般用于成节的内容，会在文档流中开始一个新的节。它用来表现普通的文档内容或应用区块，

通常由内容及其标题组成。但 section 元素标签并非一个普通的容器元素，它表示一段专题性的内容，一般会带有标题。

当描述一件具体的事物的时候，通常鼓励使用 article 来代替 section；当使用 section 时，仍然可以使用 h1 来作为标题，而不用担心它所处的位置，以及其他地方是否用到；当一个容器需要被直接定义样式或通过脚本定义行为时，推荐使用 DIV 元素而非 section。<section>标签示例代码如下。

```
<section>
<h1>section 是什么?</h1>
<h2>一个新的章节</h2>
<article>
<h2>关于 section</h1>
<p>section 的介绍</p>
…
</article>
</section>
```

2.3.2 article 标签

<article>是一个特殊的 section 标签，它比 section 具有更明确的语义，代表一个独立的、完整的相关内容块，可独立于页面其他内容使用。例如，一篇完整的论坛帖子，一篇博客文章，一个用户评论等。一般来说，article 会有标题部分(通常包含在 header 内)，有时也会包含 footer。article 可以嵌套，内层的 article 对外层的 article 标签有隶属关系。例如，一篇博客的文章，可以用 article 显示，然后一些评论可以以 article 的形式嵌入其中。<article>标签示例代码如下。

```
<article>
<header>
<hgroup>
<h1>这是一篇介绍 HTML5 结构标签的文章</h1>
<h2>HTML5 的革新</h2>
</hgroup>
<time datetime="2015-08-20">2015.08.20</time>
</header>
<p>文章内容详情</p>
</article>
```

2.3.3 nav 标签

nav 标签代表页面的一个部分，是一个可以作为页面导航的链接组，其中的导航元素链接到其他页面或者当前页面的其他部分，使 HTML 代码在语义化方面更加精确，同时对于屏幕阅读器等设备的支持也更好，<nav>标签示例代码如下。

```
<nav>
<ul>
```

```
<li>首页</li>
<li>新闻综合</li>
<li>联系我们</li>
</ul>
</nav>
```

2.3.4 aside 标签

aside 标签用来装载非正文的内容，被视为页面里面一个单独的部分。它包含的内容与页面的主要内容是分开的，可以被删除，而不会影响网页的内容、章节或是页面所要传达的信息。例如，广告、成组的链接、侧边栏等。<aside>标签示例代码如下。

```
<aside>
<h1>作者简介</h1>
<p>IT</p>
</aside>
```

2.3.5 header 标签

<header>标签定义文档的页眉，通常是一些引导和导航信息。它不局限于写在网页头部，也可以写在网页内容里面。

通常<header>标签至少包含(但不局限于)一个标题标记(<h1>～<h6>)，还可以包括<hgroup>标签，也可以包括表格内容、标识、搜索表单、<nav>导航等。<header>标签示例代码如下。

```
<header>
<hgroup>
<h1>网站标题</h1>
<h1>网站副标题</h1>
</hgroup>
</header>
```

2.3.6 footer 标签

footer 标签定义 section 或 document 的页脚，包含与页面、文章或是部分内容有关的信息，比如文章的作者或者日期。作为页面的页脚时，一般包含版权、相关文件和链接。它和<header>标签的使用方法基本一样，可以在一个页面中多次使用，如果在一个区段的后面加入 footer，那么它就相当于该区段的页脚了。<footer>标签示例代码如下。

```
<footer>
COPYRIGHT@ 版权所有
</footer>
```

2.3.7 hgroup 标签

hgroup 标签是对网页或区段 section 的标题元素(h1～h6)进行组合。例如，在一区段

中有连续的 h 系列的标签元素，则可以用 hgroup 将它们括起来。<hgroup>标签示例代码如下。

```
<hgroup>
<h1>这是一篇介绍 HTML 5 结构标签的文章</h1>
<h2>HTML 5 的革新</h2>
</hgroup>
```

2.3.8 figure 标签

figure 标签用于对元素进行组合，多用于图片与图片描述组合。<figure>标签示例代码如下。

```
<figure>
<img src="img.gif" alt="figure 标签"  title="figure 标签" />
<figcaption>这儿是图片的描述信息</figcaption>
</figure>
```

2.3.9 表单标签

在之前的 HTML 表单标签中，对于一些功能支持得不够好，比如：文本框提示信息（之前只能通过 JavaScript 和 input 的事件结合处理）、表单校验、日期选择控件、颜色选择控件、范围控件、进度条、标签跨表单等功能。当然可以直接通过 JavaScript 和 DOM 元素配合实现这些通用的功能。这些功能和标签都已经大量地使用在了现代的 Web 应用中，而这些公共性的东西在早期的 HTML 标准中没有直接的标准支持，而在 HTML5 中，新标准直接把这些常用的基本功能直接加入到新的表单标签中，真正把表单功能变得异常强大。新的标准中添加了很多输入型控件，比如 number、url、email、range、color 等。而它们都是以 input 标签的 type 属性出场，下面简单介绍一下。

1. 只能输入数字的 number 类型 input 标签

number 类型的 input 标签用于应该包含数值的输入域，还能够设定对所接受的数字的限定。例如，只能输入数字的 number 类型 input 标签的 HTML 代码如下。

```
<input type="number" name="demoNumber" min="1" max="100" step="2"/>
```

注：此标签其实就是普通的 input 标签，只不过是 type 类型指向了 number，标识当前标签只接受数字类型输入，另外添加了 4 个属性。

(1) name：用来标识表单提交时的 key 值。

(2) min：是表单标签新增加的属性，标识当前输入框输入的最小值。

(3) max：标识输入框输入的最大值。

(4) step：是步长的意思，也就是在增大或者减小的时候所增加或减少的步长。

小结：min、max、step 是表单标签中添加的新的属性。另外就是 type 又增加了一个新的 number 类型，接受数字输入。而之前要做到这样的效果只能通过 JavaScript 在失去焦点时候判断，控制起来不那么方便，现在一切都非常简单。

2. 新型 email 类型 input 标签

email 类型的 input 标签用于应该包含 E-mail 地址的输入域。在提交表单时，会自动验证 email 域的值。例如，新型 email 类型 input 标签的 HTML 代码如下。

```
<input type = "email" name = "email" placeholder = "请输入注册邮箱" />
```

在上面的 HTML 代码中，相对于之前的标签，不同点是：type＝"email"表示当前 input 标签接收一个邮箱的输入。另外就是：placeholder＝"请输入注册邮箱"，这个属性的功能，相信看到此时的效果的时候用户会感到非常兴奋，而在之前实现此提示信息，需要监听一下文本框的 blur 事件，然后判断是否为空，为空再去给文本框赋值一个灰色的字体提示信息，而现在只需要一个简单属性指定就可以实现了。

小结：当表单在提交前，此文本框会自动校验是否符合邮箱的正则表达式，另外 placeholder 属性带来的提示信息功能也十分强大。

3. 新型 url 类型 input 标签

跟 email 类型 input 标签类似，url 类型用于应该包含 URL 地址的输入域。在提交表单时，会自动验证 url 域的值。例如：

```
Homepage: <input type = "url" name = "user_url" />
```

4. 新型 tel 类型 input 标签

也跟 email 类型 input 标签类似，tel 类型用于应该包含电话号码的输入域。示例代码如下。

```
<input type = "tel" placeholder = "输入电话" name = "phone"/>
```

5. 新型 range 类型 input 标签

此类型标签的加入，输入范围内的数据变得非常简单容易，而且非常标准，用户输入可感知体验非常好。另外，此标签可以跟表单新增加的 output 标签一块使用，达到一个联动的效果。请看如下的代码。

```
<form action = "" method = "POST" id = "form1">
  <input type = "range" min = "0" max = "50" step = "5" name = "rangedemo" value = "0" />
  <output onforminput = "value = rangedemo.value">0</output>
</form>
```

6. 新的日期、时间、月份、星期 input 标签

如果做过相关的 Web 项目开发，肯定会遇到相关的 JavaScript 日期控件。而一系列版本问题，使用不方便等问题将一去不复返，示例代码如下。

```
<input type = "date" name = "datedemo" />
```

当然还有其他的 type，比如 month(月)、time、week、datetime-local、datetime。

7. 新的日期、时间、月份、星期 input 标签

在之前版本的 HTML 中，颜色选择器实现起来还是有点儿困难的，而现在一切都变得那么简单。例如下面的代码。

```
<input type="color"/>
```

8. input 标签自动完成功能

如果用户有一定的开发经验，肯定做过相关的自动完成或者输入提示功能。那这将不再那么复杂，简单几个标签就可自动实现此功能，请看如下代码。

```
<input type="text" autocomplete="on" name="demoAutoComplete" list="autoNames" />
<datalist id="autoNames">
    <option  value="Web 开发" ></option>
    <option  value="手机开发" ></option>
    <option  value="算法实现" ></option>
</datalist>
```

9. 表单新增属性

在 HTML5 中，表单新增加的属性如下。

1）autocomplete 属性

autocomplete 属性可以应用于<form>标签或者是以下<input>标签：color、date、email、password、range、search、telephone、text 或 url。该属性的有效参数是“on”（默认）和“off”。

当 autocomplete 属性设置为“on”时，先前已经获得输入的任何字段都会记得之前输入的值，并将这些值作为建议提供给用户，用户不必再次进行输入。

当 autocomplete 属性设置为“off”时，该功能是被禁止的。在<form>标签中应用该属性时，表单中的所有相关字段都会受到影响。当应用于一个<input>标签时，则只会影响<input>标签对应的字段。下面是使用该属性的例子。

```
<form action="prog.php"  method="post"  autocomplete="on">
  <input type="text" name="field1" autocomplete="off"/>
</form>
```

所有主流浏览器的最新版本都支持 autocomplete 属性。

2）autofocus 属性

autofocus 属性可以应用于任何<input>标签，在页面加载时自动获得输入焦点。这与将光标放置于输入字段之上具有相同效果，可以方便用户输入，或者选择任何其他类型的字段为用户的操作做准备，该属性的激活方式如下所示。

```
<input type="text" name="field" autofocus="autofocus"/>
```

除了 IE，所有主流浏览器的最新版本都支持 autofocus 特性，但可以在网页中安全地使用 autofocus 属性，因为 IE 仍然可以运行得很好——只是不提供自动获得任何字段焦点的

功能而已。

3）form 属性

应用 form 属性就没有必要在表单中放置<input>标签。只要赋予表单一个 ID，就可以将该表单指定为 form 属性的参数。

例如，下面的代码打开然后关闭 id 为 form1 的表单，在该表单后<input>标签链接了该表单。

```
<form id="form1" action="prog.php" method="post">
    <input type="submit"/>
</form>
<input type="text" name="field" form="form1"/>
```

该特性目前仅在 Opera 浏览器中可用，因此在其他主流浏览器能够支持该特性之前建议不要使用。

4）formaction 属性

formaction 属性是一种表单重载，可以使用它将表单的 action 属性改为不同的目标。

例如在下面的代码中，表单不会发送至<form>标签中指定的 prog.php 程序，而是发送至 prog2.php。

```
<form action="prog.php" method="post">
  <input type="text" name="field"/>
  <input type="submit" formaction="prog2.php"/>
</form>
```

当希望提供不止一个提交按钮时，formaction 属性特别有用，这样需要提交至不同目标的表单都可以和不同程序相关联。

表单重载可以在 submit 或者 image 类型的<input>标签上运行，但目前只有 Opera 浏览器支持该特性，因此在其他主要浏览器支持该特性之前建议不要使用。

5）formenctype 属性

formenctype 属性是一种可以改变表单的编码类型（enctype 属性）的表单重载，与 formaction 使用方式相似（目前仅 Opera 支持该属性，因此目前不建议使用）。

6）formmethod 属性

formmethod 属性是一种可以改变传送方法（method 属性的参数“post”或者“get”）的表单重载，使用方法与 formaction 相似（该属性目前只能在 Opera 上运行）。

7）formnovalidate 属性

formnovalidate 属性是一种表单重载，可以使用 formnovalidate 属性改变 novalidate 属性的值，使用方法与 formaction 相似（该属性目前只能在 Opera 上运行）。

8）formtarget 属性

formtarget 属性是一种可以改变 target 属性值的表单重载，使用方法与 formaction 相似（该属性目前只能在 Opera 上运行）。

9）height 属性和 width 属性

height 属性和 width 属性可以应用于 image 类型的<input>标签，用来改变图像的高

度和宽度。这两个属性可以在所有最新版本的主要浏览器上使用,使用方法如下。

```
<input type="image" src="image.png" height="20" width="60"/>
```

10) list 属性、<datalist>标签和<option>标签

有些输入字段支持列表,可以使用 list 属性来引用这些输入字段。例如,下面的代码使用 list 属性与新的<datalist>标签一起提供一个供选择的 URL 组合。

```
Choose a Web page: <input type="url" name="links" list="sites" />
<datalist id="links">
<option label="Google" value="http://google.com" />
<option label="Yahoo! " value="http://yahoo.com" />
<option label="Bing" value="http://bing.com" />
<option label="Ask" value="http://ask.com" />
</datalist>
```

该特性与 autocomplete 属性的运行方式相似,不同之处是当输入获得焦点时将会显示用户定义的建议选择列表。目前仅 Opera 浏览器支持该特性,但是用户仍然可以在网页中使用,因为其他浏览器根本无法显示建议列表。随着时间的推移,当其他浏览器都支持该属性时使用该特性的 Web 表单将会变得更快,能够满足大多数访问者的要求。

11) min 属性

min 属性用来指定包含数字或者日期的输入类型的最小值。不建议依赖于该类型的验证,因为只有 Chrome 和 Opera 浏览器支持该属性,下面是使用 min 属性的一个示例。

```
<input type="time" name="deliver" value="09:00" min="09:00"/>
```

12) max 属性

max 属性用来指定包含数字或者日期的输入类型的最大值,目前只有 Chrome 浏览器和 Opera 浏览器支持 max 属性。可以将 min 属性和 max 属性组合使用,如下所示。

```
<input type="time" name="deliver" value="09:00" min="09:00" max="17:00"/>
```

13) multiple 属性

multiple 属性允许用户在 email 或 file 类型的<input>标签中选择多个值。除了 Internet Explorer 和 Opera,该属性可以在所有主要浏览器的最新版本上运行。使用方式如下。

```
<input type="file" name="images" multiple="multiple"/>
```

然后,当浏览对话框弹出时,可以同时选择多个文件(通常配合使用 Ctrl 键)。而在不支持该特性的浏览器上一次只能选择一个文件。

14) novalidate 属性

novalidate 属性指定某个表单在提交时不应得到验证。novalidate 属性的值可以是“true”或者“false”。目前所有主流浏览器都不支持 novalidate 属性,因此不能使用。在支持

novalidate 属性的浏览器上，该属性可以应用于<form>标签以及如下类型的<input>标签：color、date、email、password、range、search、telephone、text 或 url。使用方式如下。

```
< input type = "text" name = "field" novalidate = "true"/>
```

一旦该属性得到支持，用户随时都可以使用它，至少 HTML5 的验证特性要优于目前能够提供的验证。

15）pattern 属性

可以使用 pattern 属性在某个输入字段中指定一个正则表达式，之后将会计算该输入字段的值。该属性可以应用于下面类型的<input>标签：email、password、search、telephone、text 或 url。例如，如果在某个字段中只允许输入字母数字字符、连字符、下划线字符，那么可以使用下面的 HTML。

```
< input type = "text" name = "username" pattern = "[ \w\ - ] { 6, 16}"/>
```

格式“[\w\-_]{6,16}”告诉浏览器只接受下面的字符。

(1) \w：a～z 和 A～Z 的字母、0～9 的数字和下划线。

(2) \-：连字符。

(3) {6,16}：字符长度在 6～16 之间。

然而，目前仅 Opera 浏览器和 Chrome 浏览器支持该特性。

16）placeholder 属性

placeholder 属性可以在一个空白输入字段上放置有用的提示，使用该属性可以帮助用户输入有效的内容。使用方式如下。

```
< input type = "text" name = "username" size = "35" placeholder = "Enter your 6 - 16 character
username"/>
```

size 属性的值为 35，可以确保占位符文本拥有足够的空间。在向字段中输入内容之前以浅色显示占位符文本。只要字段获得焦点，提示消息将会消失，准备输入文本。该属性可以应用于以下类型的<input>标签：email、password、search、telephone、text 和 url。

placeholder 属性可以在所有主流浏览器的最新版本上使用，除了 IE 和 Opera。但是该属性可以安全使用，因为 IE 和 Opera 会简单地忽略该属性。

17）required 属性

required 属性用来确保在表单提交之前字段已经输入完毕，使用方式如下。

```
< input type = "number" name = "age" required = "required"/>
```

为了使 required 属性正常运行，网页的“!DOCTYPE”必须使用“<!DOCTYPE HTML>”(而不是使用任何的 HTML4.01 文档类型)。

与许多其他的验证特性一样，目前只有 Chrome 和 Opera Web 浏览器支持该特性，因此对所有 Web 表单的浏览器内部验证来说，required 属性是不可靠的。

18）step 属性

step 属性用来为包含数字或者日期的输入类型指定步长值，目前只有 Chrome 浏览器和 Opera 浏览器支持该属性。下面是将 step 属性与 min 和 max 属性组合使用的示例。

```
< input type = "time" name = "deliver" value = "09: 00" min = "09: 00" max = "17: 00" step = "3600" />
```

step 属性的值可以是任何正整数，上例中 step 值的单位是时间秒。

2.4 HTML5 文件操作

以前操作本地文件都是使用 Flash、Silverlight 或者第三方的 ActiveX 插件等技术，由于使用了这些技术后就很难进行跨平台或者跨浏览器、跨设备等情况下实现统一的表现，从另外一个角度来说就是让 Web 应用依赖了第三方的插件，而不是很独立，不够通用。在 HTML5 标准中，默认提供了操作文件的 API 让这一切直接标准化。有了操作文件的 API，Web 应用就可以很轻松地通过 JavaScript 来控制文件的读取、写入、文件夹、文件等一系列的操作，让 Web 应用不再那么蹩脚，而之前 Web 应用如果不借助第三方插件，那基本上难以实现。但是最新的标准中大部分浏览器都已经实现了文件的读取 API，文件的写入，文件和文件夹的最新的标准刚制定完毕，相信后面随着浏览器的升级这些功能肯定会实现得非常好，接下来主要介绍文件读取的几个 API。首先介绍几个重要的 JavaScript 对象。

2.4.1 FileList 对象

它是 File 对象的一个集合，在 HTML4 标准中文件上传控件只接受一个文件，而在新标准中，只需要设置 multiple 属性，就可以支持多文件上传，所以从此标签中获取的 files 属性就是 FileList 对象实例。例如：<input type＝"file" multiple＝"multiple" name＝"fileDemo" id＝"fileDemo" />。下面是关于 FileList 对象的 API 的原型。

```
interface FileList {
      getter File? item(unsigned long index);
      readonly attribute unsigned long length;
};
```

2.4.2 Blob 对象

Blob 对象其实就是一个原始数据对象，它提供了 slice 方法可以读取原始数据中的某块数据。另外有两个属性：size（数据的大小）、type（数据的 MIME 类型）。下面的代码是 W3C 的 API 原型。

```
interface Blob {
  readonly attribute unsigned long long size;
  readonly attribute DOMString type;
  Blob slice(optional long long start,
             optional long long end,
             optional DOMString contentType);

};
```

2.4.3 File 对象

File 对象继承自 Blob 对象，指向一个具体的文件，它还有两个属性：name（文件名）、lastModifiedDate（最后修改时间）。下面看一下 W3C 的标准。

```
interface File: Blob {
        readonly attribute DOMString name;
        readonly attribute Date lastModifiedDate;
};
```

2.4.4 FileReader 对象

FileReader 对象用来读取文件里面的数据，提供三个常用的读取文件数据的方法，另外读取文件数据使用了异步的方式，非常高效。下面看一些 W3C 的标准。

```
[Constructor]
    interface FileReader: EventTarget {
      //异步读取方法
      void readAsArrayBuffer(Blob blob);
      void readAsBinaryString(Blob blob);
      void readAsText(Blob blob, optional DOMString encoding);
      void readAsDataURL(Blob blob);
      void abort();
      //状态
      const unsigned short EMPTY  =  0;
      const unsigned short LOADING  =  1;
      const unsigned short DONE  =  2;
      readonly attribute unsigned short readyState;
      //文件或块数据
      readonly attribute any result;
      readonly attribute DOMError error;
      //事件处理属性
      attribute [TreatNonCallableAsNull] Function   onloadstart;
      attribute [TreatNonCallableAsNull] Function   onprogress;
      attribute [TreatNonCallableAsNull] Function   onload;
      attribute [TreatNonCallableAsNull] Function   onabort;
      attribute [TreatNonCallableAsNull] Function   onerror;
      attribute [TreatNonCallableAsNull] Function   onloadend;
};
```

这个对象是非常重要的一个对象，它提供了 4 个读取文件数据的方法，这些方法都是以异步的方式读取数据，读取成功后就直接将结果放到属性 result 中。所以一般就是直接读取数据，然后监听此对象的 onload 事件，然后在事件里面读取 result 属性，再做后续处理。当然 abort 就是停止读取的方法。其他的就是事件和状态，不再赘述。

对上面三个方法介绍如下。

readAsBinaryString(Blob blob)：传入一个 Blob 对象，然后读取数据的结果作为二进制字符串的形式放到 FileReader 的 result 属性中。

readAsText(Blob blob, optional DOMString encoding):第一个参数传入 Blog 对象,然后第二个参数传入编码格式,异步将数据读取成功后放到 result 属性中,读取的内容是普通的文本字符串的形式。

readAsDataURL(Blob blob):传入一个 Blob 对象,读取内容可以作为 URL 属性,也就是说可以将一个图片的结果指向一个 img 的 src 属性。

2.4.5 文件操作实例

以前操作一个图片文件,都是先将图片上传到服务器端,然后再使用一个 img 标签指向服务器的 URL 地址,再使用第三方插件进行图片处理,而现在这一切都不需要服务器端了,因为 FileReader 对象提供的几个读取文件的方法变得异常简单,而且全不是客户端 JavaScript 的操作。请看下面读取上传文件内容,然后将文件内容直接读取到浏览器上的示例代码。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title></title>
        <script src="Scripts/jquery-1.5.1.js" type="text/javascript"></script>
        <script type="text/javascript">

            if(typeof FileReader == "undified") {
                alert("您的浏览器不行了!");
            }

            function showDataByURL() {
                var resultFile = document.getElementById("fileDemo").files[0];
                if (resultFile) {
                    var reader = new FileReader();

                    reader.readAsDataURL(resultFile);
                    reader.onload = function (e) {
                      var urlData = this.result;
                      document.getElementById("result").innerHTML += "<img src=" +
                      urlData + " alt=" + resultFile.name + "/>";
                    };

                }

            }

            function showDataByBinaryString() {
                 var resultFile = document.getElementById("fileDemo").files[0];
               if (resultFile) {
                     var reader = new FileReader();
                     //异步方式,不会影响主线程
```

```
                    reader.readAsBinaryString(resultFile);

                    reader.onload = function(e) {
                        var urlData = this.result;
                        document.getElementById("result").innerHTML += urlData;
                    };
                }
            }


            function showDataByText() {
                var resultFile = document.getElementById("fileDemo").files[0];
                if (resultFile) {
                    var reader = new FileReader();
                    reader.readAsText(resultFile,"gb2312");
                    reader.onload = function (e) {
                        var urlData = this.result;
                        document.getElementById("result").innerHTML += urlData;
                    };
                }
            }

        </script>
    </head>
    <body>
        <input type="file" name="fileDemo" id="fileDemo" multiple="multiple"/>
        <input type="button" value="readAsDataURL" id="readAsDataURL"
        onclick="showDataByURL();"/>
        <input type="button" value="readAsBinaryString" id="readAsBinaryString"
        onclick="showDataByBinaryString();"/>
        <input type="button" value="readAsText" id="readAsText"
        onclick="showDataByText();"/>
        <div id="result">
        </div>
    </body>
    </html>
```

总之,有了文件操作的 API 后,让 JavaScript 进一步地操作本地文件的功能得到空前的加强,HTML5 对于客户端 Web 应用得到进一步功能的提升,HTML5 的趋势让 Web 更加富客户端化,而这些都需要让 HTML 或者 JavaScript 变得更加强大。

2.5 画 布

Canvas 的概念最初是由苹果公司提出的,用于在 Mac OS X Webkit 中创建控制板部件。在 Canvas 出现之前,开发人员若要在浏览器中使用绘图 API,只能使用 Adobe 的 Flash 和 SVG(Scalable Vector Graphics,可伸缩矢量图形)插件,或者只有 IE 才支持的 VML(Vector Markup Language,矢量标记语言),以及其他一些稀奇古怪的 JavaScript

技巧。

假设要在没有 canvas 元素的条件下绘制一条对角线——听起来似乎很简单,但实际上如果没有一套二维绘图 API 的话,这会是一项相当复杂的工作。HTML5 中的 Canvas 能够提供这样的功能,对浏览器端来说此功能非常有用,因此 Canvas 被纳入了 HTML5 规范。

起初,苹果公司曾暗示可能会为 WHATWG(Web Hypertext Application Technology Working Group,Web 超文本应用技术工作组)草案中的 Canvas 规范申请知识产权,这在当时引起了一些 Web 标准化追随者的关注。不过,苹果公司最终还是按照 W3C 的免版税专利权许可条款公开了其专利。

在网页上使用 canvas 元素时,它会创建一块矩形区域。默认情况下,该矩形区域宽为 300px,高为 150px,但也可以自定义具体的大小或者设置 canvas 元素的其他特性。

在页面中加入了 canvas 元素后,就可以通过 JavaScript 来自由地控制它。可以在其中添加图片、线条以及文字,也可以在里面绘图,甚至还可以加入高级动画。

大多数主流操作系统和框架支持的二维绘制操作,HTML5 Canvas API 都支持。如果曾经有过二维图像编程的经验,那么会对 HTML5 Canvas API 感觉非常顺手,因为这个 API 就是参照既有系统设计的。如果没有这方面的经验,则会发现与这么多年来一直使用的图片加 CSS 开发 Web 图形的方式比起来,Canvas 的渲染系统有多么强大。

2.5.1 Canvas 重要 Context 对象

(1) 要使用 Canvas 来绘制图形必须在页面中添加 Canvas 的标签,例如:

```
<canvas id = "demoCanvas" width = "500" height = "500"><p>现在都用 html5 </p> </canvas>
```

(2) 当然只有上面的标签,只能是创建好了一个画布,其中,width 和 height 属性就是设置画布的大小。id 属性也是必需的,后面要用 id 来拿到当前的 Canvas 的 DOM 对象。通过此 Canvas 的 DOM 对象就可以获取它的上下文了,Canvas 绘制图形都是依靠 Canvas 对象的上下文对象。例如:

```
<script type = "text/javascript">
//第一步: 获取 canvas 元素
var canvasDom = document.getElementById("demoCanvas");
//第二步: 获取上下文
var context = canvasDom.getContext("2d");
</script>
```

(3) Context 上下文默认有两种绘制方式:绘制线(stroke)、填充(fill)。注意:决定了使用哪种方式之后,在填充或者绘制线之前应先设置样式。

2.5.2 Canvas 绘制的步骤

Canvas 绘制的总体步骤如下。

(1) 创建 HTML 页面,设置画布标签。

(2) 编写 JavaScript,获取画布 DOM 对象。

(3) 通过 Canvas 标签的 DOM 对象获取上下文。

(4) 设置绘制线样式、颜色。

(5) 绘制矩形,或者填充矩形。

下面以 Canvas 绘制一个矩形和填充一个矩形为例,说明如何使用 Canvas。

```
<body>
    <canvas id="demoCanvas" width="500" height="500">
        <p>html5 时代了</p>
    </canvas>
    <!--- 下面将演示一种绘制矩形的 demo --->
    <script type="text/javascript">

        //第一步: 获取 canvas 元素
        var canvasDom = document.getElementById("demoCanvas");
        //第二步: 获取上下文
        var context = canvasDom.getContext("2d");
        //第三步: 指定绘制线样式、颜色
        context.strokeStyle = "red";
        //第四步: 绘制矩形,只有线.内容是空的
        context.strokeRect(10, 10, 190, 100);

        //以下演示填充矩形.
        context.fillStyle = "blue";
        context.fillRect(110,110,100,100);
    </script>
</body>
```

显示效果如图 2.3 所示。

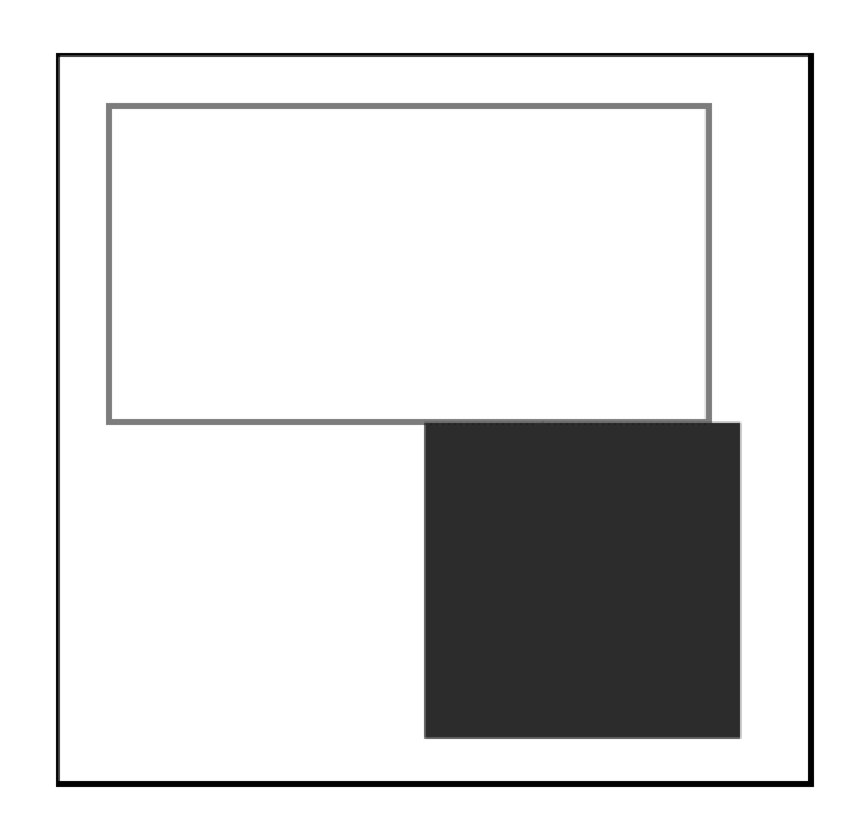


图 2.3　绘制矩形效果图

2.5.3　Canvas 绘制线条

Context 对象的 beginPath 方法表示开始绘制路径,moveTo(x,y)方法设置线段的起点,lineTo(x,y)方法设置线段的终点,stroke 方法用来给透明的线段着色。moveTo 和

lineTo 方法可以多次使用。最后,还可以使用 closePath 方法,自动绘制一条当前点到起点的直线,形成一个封闭图形,减少使用一次 lineTo 方法,示例代码如下。

```
<body>
   <canvas id="demoCanvas" width="500" height="600">
   </canvas>
   <script type="text/ javascript">
      //通过 id 获得当前的 Canvas 对象
      var canvasDom = document.getElementById("demoCanvas");
      //通过 Canvas DOM 对象获取 Context 的对象
      var context = canvasDom.getContext("2d");
      context.beginPath();                    //开始路径绘制
      context.moveTo(20, 20);                 //设置路径起点,坐标为(20,20)
      context.lineTo(200, 200);               //绘制一条到(200,20)的直线
      context.lineTo(400, 20);
      context.closePath();
      context.lineWidth = 2.0;                //设置线宽
      context.strokeStyle = "#CC0000";        //设置线的颜色
      context.stroke();                       //进行线的着色,这时整条线才变得可见
  </script>
</body>
```

2.5.4 Canvas 绘制文本

Context 上下文对象的 fillText(string,x,y)方法是用来绘制文本,它的三个参数分别为文本内容、起点的 x 坐标、y 坐标。使用之前,需用 font 设置字体、大小、样式(写法类似于 CSS 的 font 属性)。与此类似的还有 strokeText 方法,用来添加空心字。另外注意一点:fillText 方法不支持文本断行,即所有文本出现在一行内。所以,如果要生成多行文本,只有多次调用 fillText 方法,示例代码如下。

```
<canvas id="demoCanvas" width="500" height="600"></canvas>
<script type="text/javascript">
  //通过 id 获得当前的 Canvas 对象
  var canvasDom = document.getElementById("demoCanvas");
  //通过 Canvas DOM 对象获取 Context 的对象
  var context = canvasDom.getContext("2d");
  context.moveTo(200,200);
//设置字体
  context.font = "Bold 50px Arial";
  //设置对齐方式
  context.textAlign = "left";
  //设置填充颜色
  context.fillStyle = "#005600";
//设置字体内容,以及在画布上的位置
   context.fillText("老马!", 10, 50);
  //绘制空心字
  context.strokeText("blog.itjeek.com!", 10, 100);
</script>
```

2.5.5 Canvas 绘制圆形和椭圆

Context 上下文的 arc 方法用于绘制圆形或者椭圆。arc 方法的 x 和 y 参数是圆心坐标，radius 是半径，startAngle 和 endAngle 是扇形的起始角度和终止角度（以弧度表示），anticlockwise 表示作图时应该逆时针画（true）还是顺时针画（false），示例代码如下。

```
<canvas id="demoCanvas" width="500" height="600"></canvas>
  <script type="text/javascript">
  //通过 id 获得当前的 Canvas 对象
  var canvasDom = document.getElementById("demoCanvas");
  //通过 Canvas DOM 对象获取 Context 的对象
  var context = canvasDom.getContext("2d");
  context.beginPath();                    //开始绘制路径
  //绘制以 (60,60)为圆心,50 为半径长度,从 0°到 360°(PI 是 180°),最后一个参数代表顺时针旋转
   context.arc(60, 60, 50, 0, Math.PI * 2, true);
   context.lineWidth = 2.0;               //线的宽度
   context.strokeStyle = "#000";          //线的样式
   context.stroke();                      //绘制空心的,当然如果使用 fill 那就是填充了
</script>
```

2.5.6 Canvas 绘制图片

Canvas 绘制图片是它的一大特点。当然配合 File 的 API，让 JavaScript 变得无可匹敌。下面演示一下怎样绘制图片，并且制作出一个立体效果出来，示例代码如下。

```
<canvas id="demoCanvas" width="500" height="600"></canvas>
<script type="text/javascript">
    //通过 id 获得当前的 Canvas 对象
    var canvasDom = document.getElementById("demoCanvas");
    //通过 Canvas DOM 对象获取 Context 的对象
    var context = canvasDom.getContext("2d");
    var image = new Image();              //创建一张图片
    image.src = "Images/a.png";           //设置图片的路径
    image.onload = function() {           //当图片加载完成后
    for (var i = 0; i < 10; i++) {        //输出 10 张照片
    //参数: (1)绘制的图片  (2)坐标 x,(3)坐标 y
         context.drawImage(image, 100 + i * 80, 100 + i * 80);
      }
    };
    </script>
```

上面介绍了 Canvas 最基本的用法，总体来说，使用 Canvas 绘制图片和文本、形状都不是很难，API 代码上手难度基本就是初级。但是由此而带来的非常多的可能，让 HTML5 真正得强大到无可比拟的地步。当然本文并没有涉及 Canvas 3D 绘制的相关内容，而且关于 Canvas 绘制渐变色、绘制阴影、图片的相关处理操作等，这些内容如果读者确实需要，可以直接阅读 HTML5 的最新标准文档。

2.6 地理位置

地理位置(Geolocation)是 HTML5 的重要特性之一,提供了确定用户位置的功能,借助这个特性能够开发基于位置信息的应用。HTML5 的地理位置特性可以返回网页访问者的地理位置(或最接近的位置),它允许用户在 Web 应用程序中共享位置,使其能够享受位置感知服务,HTML5 中使用 getCurrentPosition()方法来获得用户的位置。HTML5 Geolocation 规范提供了一套保护用户隐私的机制,必须先得到用户明确许可,才能获取用户的位置信息。不过,从可接触到的 HTML5 Geolocation 应用程序示例中可以看到,通常会鼓励用户共享这些信息。

本节将首先介绍 HTML5 Geolocation 位置信息的构成:纬度、经度,以及获得这些数据的途径(GPS、WiFi 和蜂窝站点)。然后将讨论 HTML5 地理定位数据的隐私问题,以及浏览器如何使用这些数据。最后将探讨它在实际中的应用,演示并帮助读者学会如何使用它构建一个实用的基于地理位置的 Web 应用

在 HTML5 中,当请求一个位置信息时,如果用户同意,浏览器就会返回位置信息,该位置信息是通过支持地理定位功能的底层设备(比如笔记本或手机)提供给浏览器的。位置信息由纬度、经度坐标和一些其他元数据组成。例如,北京故宫的位置信息主要由一对纬度和经度坐标组成:北纬 39.9°,东经 116.4°。

经纬度坐标有两种表示方式:十进制格式(例如 39.9)和 DMS(Degree Minute Second,角度)格式(例如 39°54′20″)。HTML5 Geolocation API 返回的坐标格式为十进制格式。除了纬度和经度坐标,HTML5 Geolocation 还提供位置坐标的准确度。除此之外,它还会提供其他一些元数据,比如海拔、海拔准确度、行驶方向和速度等,具体情况取决于浏览器所在的硬件设备。位置信息一般从如下数据源获得:IP 地址、三维坐标、GPS(Global Positioning System,全球定位系统)、WiFi、手机信号、用户自定义数据。它们各有优缺点,如表 2.1 所示,为了保证更高的准确度,许多设备使用多个数据源组合的方式。

表 2.1 数据源

数据源	优　点	缺　点
IP 地址	任何地方都可用,在服务器端处理	不精确(经常出错,一般精确到城市级),运算代价大
GPS	很精确	定位时间长,耗电量大,室内效果差,需要额外硬件设备支持
WiFi	精确,可在室内使用,简单、快捷	在乡村这些 WiFi 接入点少的地区无法使用
手机信号	相当准确,可在室内使用,简单、快捷	需要能够访问手机或其 Modem 设备
用户自定义	可获得比程序定位服务更准确的位置数据,用户自行输入可能比自动检测更快	可能很不准确,特别是当用户位置变更后

2.6.1 浏览器支持情况

各浏览器对 HTML5 Geolocation 的支持程度不同,并且还在不断更新中。好消息是:在 HTML5 的所有功能中,HTML5 Geolocation 是第一批被全部接受和实现的功能之一,

相关规范已经达到一个非常成熟的阶段,不大可能做太大改变。如表 2.2 所示,很多浏览器已经支持 HTML5 Geolocation。

表 2.2 各浏览器支持情况

浏 览 器	支 持 情 况
Firefox	3.5 及以上版本支持
Chrome	在带有 Gears 的第 2 版 Chrome 中被支持
Internet Explorer	通过 Gears 插件支持
Opera	在版本 10 中支持
Safari	在版本 4 中支持以实现在 iPhone 上可用

由于浏览器对它的支持程度不同,在使用之前最好先检查浏览器是否支持 HTML5 Geolocation API。后面将讲解如何检查浏览器是否支持此功能。

2.6.2 HTML5 Geolocation API

在调用 HTML5 Geolocation API 函数前,需要确保浏览器支持此功能。当浏览器不支持时,可以提供一些替代文本,以提示用户升级浏览器或安装插件(如 Gears)来增强现有浏览器功能。以下代码是浏览器支持性检查的一种途径。

```
function testSupport() {
    if (navigator.geolocation) {
        document.getElementById("support").innerHTML = "支持 HTML5 Geolocation.";
    } else {
    document.getElementById("support").innerHTML =
    "该浏览器不支持 HTML5 Geolocation,建议升级浏览器或安装插件(如 Gears).";
    }
}
```

在以上代码中,testSupport()函数检测了浏览器的支持情况。这个函数应该在页面加载的时候就被调用,如果浏览器支持 HTML5 Geolocation,navigator.geolocation 调用将返回该对象,否则将触发错误。预先定义的 support 元素会根据检测结果显示浏览器支持情况的提示信息。

在 HTML5 Geolocation 功能中,位置请求有两种:单次位置请求和重复性位置更新请求。

单次位置请求:在许多应用中,只检索或请求一次用户位置即可。例如,午餐时间到了,要查询用户附近餐馆的特色菜及其价格和评论,就可以使用如下代码所示的 HTML5 Geolocation API。

```
void getCurrentPosition(updateLocation, optional handleLocationError,optional options);
```

这个函数接受一个必选参数和两个可选参数,必选参数 updateLocation 为浏览器指明位置数据可用时应调用的函数。获取位置操作可能需要较长时间才能完成,用户不希望在检索位置时浏览器被锁定,这个参数就是异步收到实际位置信息后,进行数据处理的地方。

它同时作为一个函数，只接受一个参数：位置对象 position。这个对象包含坐标(coords)和一个获取位置数据时的时间戳，许多重要的位置数据都包含在 coords 中，比如：latitude(纬度)、longitude(经度)、accuracy(准确度)。

毫无疑问，这三个数据是最重要的位置数据。latitude 和 longitude 包含 HTML5 Geolocation 服务测定的十进制用户位置。accuracy 以 m 为单位制定纬度和经度值与实际位置间的差距。局限于 HTML5 Geolocation 的实现方式，位置只能是粗略的近似值。在呈现返回值前请一定要检查返回值的准确度。如果推荐的所谓“附近的”餐馆，实际上要耗费用户几个小时的路程，那就不好了。

坐标可能还包含其他一些数据，不能保证浏览器对其都支持，如果不支持则返回 null。

(1) latitude——海拔高度，以 m 为单位。

(2) latitudeAccuracy——海拔高度的准确度，以 m 为单位。

(3) heading——行进方向，相对于正北而言。

(4) speed——速度，以 m/s 为单位。

以下代码给出了 updateLocation()函数的常用实现代码，该函数根据坐标信息执行具体的更新操作：用获得的位置信息分别更新 HTML 页面上三个空间元素的文本。

```
function updateLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var accuracy = position.coords.accuracy;

    document.getElementById("纬度").innerHTML = latitude;
    document.getElementById("经度").innerHTML = longitude;
    document.getElementById("准确度").innerHTML = accuracy + "米";
}
```

可选参数 handleLocationError 为浏览器指明出错处理函数。位置信息请求可能因为一些不可控因素失败，这时需要在这个函数中提供对用户的解释。幸运的是，该 API 已经定义了所有需要处理的错误情况的错误编号。错误编号 code 设置在错误对象中，错误对象作为 error 参数传递给错误处理程序。这些错误编号有：

(1) UNKNOWN_ERROR (0)：不包括在其他错误编号中的错误，需要通过 message 参数查找错误的详细信息。

(2) PERMISSION_DENIED (1)：用户拒绝浏览器获得其位置信息。

(3) POSITION_UNAVAILABLE (2)：尝试获取用户信息失败。

(4) TIMEOUT (3)：在 options 对象中设置了 timeout 值，尝试获取用户位置超时。

在这些情况下，可以通知用户应用程序运行出了什么问题，如以下代码所示。

```
function handleLocationError(error) {
    switch (error.code) {
    case 0:
        updateStatus("尝试获取您的位置信息时发生错误: " + error.message);
        break;
```

```
        case 1:
            updateStatus("用户拒绝了获取位置信息请求。");
            break;
        case 2:
            updateStatus("浏览器无法获取您的位置信息。");
            break;
        case 3:
            updateStatus("获取您位置信息超时。");
            break;
    }
}
```

可选参数 options 对象可以调整 HTML5 Geolocation 服务的数据收集方式。该对象有以下三个可选参数。

(1) enableHighAccuracy——如果启动该参数,浏览器会启动 HTML5 Geolocation 服务的高精确度模式,这将导致机器花费更多的时间和资源来确定位置,应谨慎使用。默认值为 false。

(2) timeout——单位为 ms,告诉浏览器获取当前位置信息所允许的最长时间。如果在这个时间段内未完成,就会调用错误处理程序。默认值为 Infinity,即无穷大(无限制)。

(3) maximumAge——以 ms 为单位,表示浏览器重新获取位置信息的时间间隔。默认值为 0,这意味着浏览器每次请求时必须立即重新计算位置。

使用可选参数 options 更新位置请求,让其包含一个使用 JSON 对象表示的可选参数,如以下代码所示。

```
navigator.geolocation.getCurrentPosition(updateLocation, handleLocationError, {timeout: 10000});
```

这个调用告诉 HTML5 Geolocation,当获取位置请求的处理时间超过 10s(10 000ms)时触发错误处理程序,这时,error code 应该是 3。

重复性位置更新请求:有时候,仅获取一次用户位置信息是不够的。比如用户正在移动,随着用户的移动,页面应该能够不断更新显示附近的餐馆信息,这样,所显示的餐馆信息才对用户有意义。幸运的是,HTML5 Geolocation 服务的设计者已经考虑到了这一点,应用程序可以使用如下 API 进行重复性位置更新请求,当监控到用户的位置发生变化时,HTML5 Geolocation 服务就会重新获取用户的位置信息,并调用 updateLocation()函数处理新的数据,及时通知用户。以下是重复性位置更新请求 API:

```
void watchPosition(updateLocation, optional handleLocationError, optional options);
```

这个函数的参数与前面提到的 getCurrentPosition 函数的参数一样,不再赘述。关闭更新也很简单,如果应用程序不需要再接收用户的位置更新消息,只需要使用 clearWatch()函数。参照下面给出的例子。

```
var watchId = navigator.geolocation.watchPosition(updateLocation,
handleLocationError);
```

```
//基于持续更新的位置信息实现一些功能…
//停止接收位置更新消息
 navigator.geolocation.clearWatch(watchId);
```

2.6.3　HTML5 Geolocation API 应用实例

下面将介绍如何用刚刚介绍的“重复性位置更新请求”构建一个简单有用的 Web 应用程序：距离跟踪器。通过此应用程序可以了解到 HTML5 Geolocation API 的强大之处。想要快速确定在一定时间内的行走距离，通常可使用 GPS 导航系统或计步器这样的专业设备。基于 HTML5 Geolocation 提供的强大服务，就可以自己创建一个网页来跟踪从网页被加载的地方到目前所在位置所经过的距离。虽然它在台式计算机上不大实用，可在手机上运行是非常理想的。只要在手机浏览器中打开这个示例页面并授予其位置访问的权限，每隔几秒钟，应用程序就会更新计算走过的距离，如图 2.4 所示。

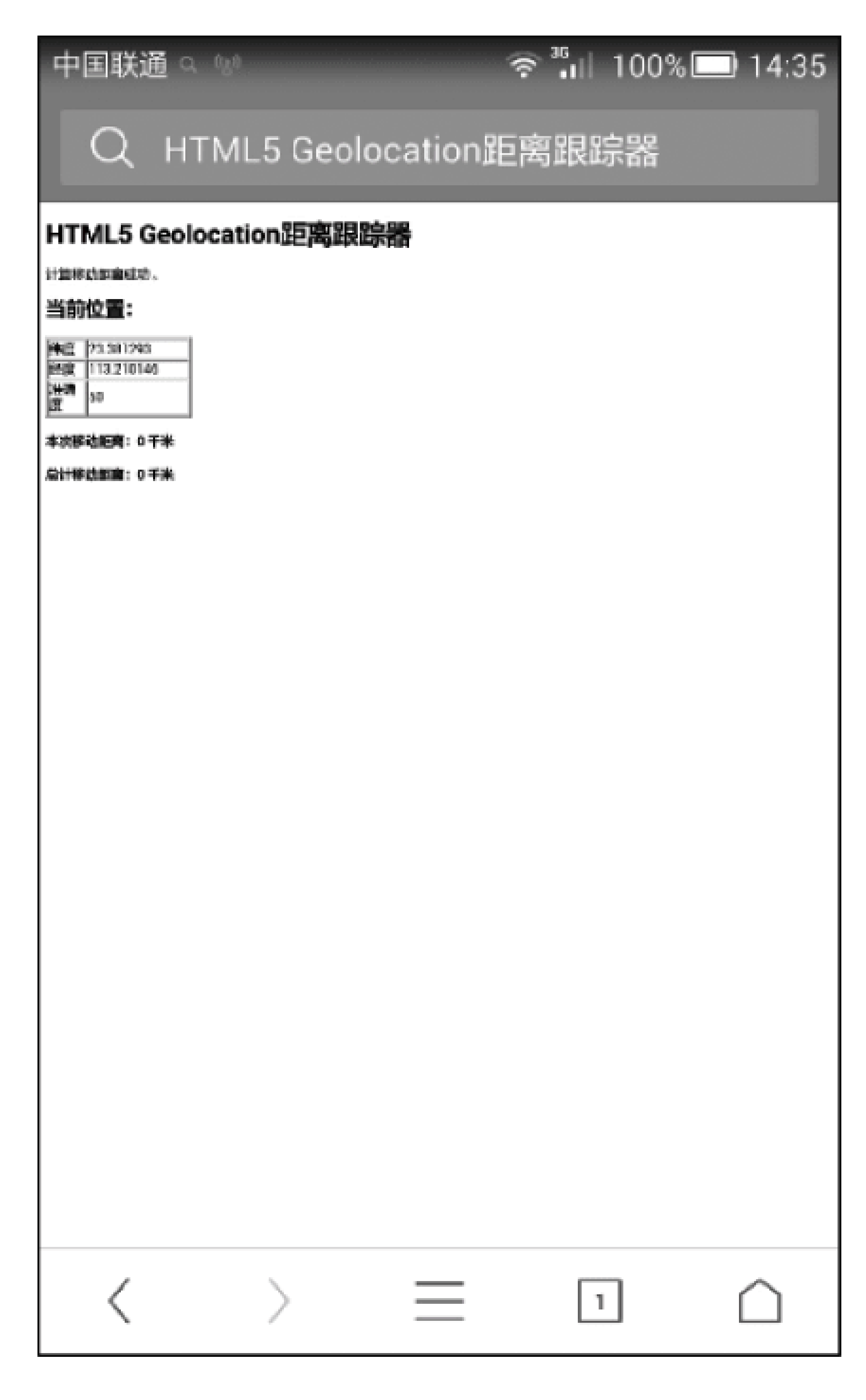


图 2.4　距离跟踪应用程序

在此实例中使用的 watchPosition()函数刚刚在 2.6.2 节中介绍过。每当有新的位置返回，就将其与上次保存的位置进行比较以计算距离。距离计算使用著名的 Haversine 公式来实现，这个公式能够根据经纬度计算地球上两点间的距离。它的 JavaScript 实现如以下代码所示。

```
function toRadians(degree) {
    return this * Math.PI / 180;
}

function distance(latitude1, longitude1, latitude2, longitude2) {
    //R 是地球半径(km)
    var R = 6371;

    var deltaLatitude = toRadians(latitude2 - latitude1);
    var deltaLongitude = toRadians(longitude2 - longitude1);
    latitude1 = toRadians(latitude1);
    latitude2 = toRadians(latitude2);

    var a = Math.sin(deltaLatitude/2) *
            Math.sin(deltaLatitude/2) +
            Math.cos(latitude1) *
            Math.cos(latitude2) *
```

```
            Math.sin(deltaLongitude/2) *
            Math.sin(deltaLongitude/2);

    var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
    var d = R * c;
    return d;
}
```

其中,distance()函数用来计算两个经纬度表示的位置间的距离,可以定期检查用户的位置,并调用这个函数来得到用户的近似移动距离。这里有一个假设,即用户在每个区间上都是直线移动的。

显示不准确的位置信息会给用户提供误导,给用户极其坏的印象,认为应用程序不可靠,应该尽量避免。因此,就可以通过 position.coords.accuracy 过滤掉所有低精度的位置更新数据。如以下代码所示。

```
//如果 accuracy 的值太大,说明它不准确,不用它计算距离
if (accuracy >= 500) {
    updateStatus("这个数据太不靠谱,需要更准确的数据来计算本次移动距离。");
    return;
}
```

最后来计算移动距离,假设前面已经至少收到了一个准确的位置,那就更新移动的总距离并显示给用户,同时还存储当前数据以备后面的比较。如以下代码所示。

```
//计算移动距离
if ((lastLat != null) && (lastLong != null)) {
    var currentDistance = distance(latitude,longitude,lastLat,lastLong);
    document.getElementById("本次移动距离").innerHTML =
        "本次移动距离:" + currentDistance.toFixed(4) + " 千米";

    totalDistance += currentDistance;

    document.getElementById("总计移动距离").innerHTML =
        "总计移动距离:" + currentDistance.toFixed(4) + " 千米";
    }

    lastLat = latitude;
    lastLong = longitude;

    updateStatus("计算移动距离成功。");
}
```

本例完整代码如下。

```
<!DOCTYPE html>
<head>
    <meta charset = "gbk">
```

```
    <title>HTML5 Geolocation 距离跟踪器</title>
</head>

<body onload="loadDemo()">

<h1>HTML5 Geolocation 距离跟踪器</h1>

<p id="status">该浏览器不支持 HTML5 Geolocation.</p>

<h2>当前位置：</h2>
<table border="1">
<tr>
<td width="40" scope="col">纬度</th>
<td width="114" id="latitude">?</td>
</tr>
<tr>
<td>经度</td>
<td id="longitude">?</td>
</tr>
<tr>
<td>准确度</td>
<td id="accuracy">?</td>
</tr>
</table>
<h4 id="currDist">本次移动距离：0 千米</h4>
<h4 id="totalDist">总计移动距离：0 千米</h4>
<script type="text/javascript">
  var totalDistance = 0.0;
  var lastLat;
  var lastLong;

function toRadians(degree) {
   return this * Math.PI / 180;
}

function distance(latitude1, longitude1, latitude2, longitude2) {
   //R 是地球半径(km)
   var R = 6371;
   var deltaLatitude = toRadians(latitude2 - latitude1);
   var deltaLongitude = toRadians(longitude2 - longitude1);
   latitude1 = toRadians(latitude1);
   latitude2 = toRadians(latitude2);
   var a = Math.sin(deltaLatitude/2) *
           Math.sin(deltaLatitude/2) +
           Math.cos(latitude1) *
           Math.cos(latitude2) *
           Math.sin(deltaLongitude/2) *
           Math.sin(deltaLongitude/2);
```

```
        var c = 2 * Math.atan2(Math.sqrt(a),Math.sqrt(1 - a));
        var d = R * c;
        return d;
    }
    function updateStatus(message) {
        document.getElementById("status").innerHTML = message;
    }

function loadDemo() {
if(navigator.geolocation) {
     updateStatus("浏览器支持 HTML5 Geolocation.");
     navigator.geolocation.watchPosition(updateLocation,handleLocationError,
                                        {maximumAge:20000});
    }
}

function updateLocation(position) {
          var latitude = position.coords.latitude;
          var longitude = position.coords.longitude;
          var accuracy = position.coords.accuracy;
          document.getElementById("latitude").innerHTML = latitude;
          document.getElementById("longitude").innerHTML = longitude;
          document.getElementById("accuracy").innerHTML = accuracy;

          //如果 accuracy 的值太大,则说明它不准确,不用它计算距离
          if (accuracy >= 500) {
           updateStatus("这个数据太不靠谱,需要更准确的数据来计算本次移动距离。);
           return;
          }

          //计算移动距离
          if ((lastLat != null) && (lastLong != null)) {
            var currentDistance = distance(latitude,longitude,lastLat, lastLong);
            document.getElementById("currDist").innerHTML =
              "本次移动距离: " + currentDistance.toFixed(4) + " 千米";
            totalDistance += currentDistance;
            document.getElementById("totalDist").innerHTML =
              "总计移动距离: " + currentDistance.toFixed(4) + " 千米";
          }

          lastLat = latitude;
          lastLong = longitude;
          updateStatus("计算移动距离成功。");
        }
    function handleLocationError(error) {
            switch(error.code)
            {
            case 0:
              updateStatus("尝试获取您的位置信息时发生错误: " + error.message);
```

```
                break;
            case 1:
                updateStatus("用户拒绝了获取位置信息请求。");
                break;
            case 2:
                updateStatus("浏览器无法获取您的位置信息: " + error.message);
                break;
            case 3:
                updateStatus("获取您位置信息超时.");
                break;
            }
    }

    </script>
    </body>
    </html>
```

至此已经构建了一个能够持续监控用户位置变化的示例应用程序，几乎完整地演示了 Geolocation API 的使用。不妨把它放到支持地理位置定位的手机或移动设备上，看看一天大概能走多少路吧。

2.7 音　　频

随着 HTML5 的出现，发生了一些重大变化，特别是在音乐和音频方面。开发人员不再要求 Web 应用程序访问者使用 Adobe Flash、Apple QuickTime 或 Microsoft Windows 媒体播放器等特殊的播放器。这意味着用户不必担心是否有最新的兼容插件(或任何播放器插件)。他们只需打开自己喜欢的浏览器并聆听能发出声音的任何内容，如音乐、有声读物或朋友和家人录制的消息。下面介绍如何利用这个最新的标准和浏览器功能。

HTML5 推出了一个简单的元素，即<audio>。使用这个标签，可以创建一个可播放声音文件的网页。如果读者在 HTML 中使用过<img>元素，会发现<audio>元素在加载外部文件方面与其类似。将<audio>元素和路径或统一资源定位符(URL)包含在音频文件里，并注明该文件是(加载页面后便)自动播放还是从播放器栏播放，这样用户可以根据需要打开或关闭声音。

浏览器制造商并非都同意使用某种音频文件格式。对于图像，PNG、JPEG 或 GIF 格式的文件在任何浏览器上都能加载到网页里。遗憾的是，音频文件并非如此。表 2.3 展示了网页中可以使用的音频文件格式，但是并非所有格式都能用于所有浏览器。例如，Chrome、Internet Explorer 9 和 Safari 浏览器不支持 WAV 文件，这是一种使用非压缩格式且正在衰败的标准。

表 2.3　音频文件格式

音频格式	Chrome	Firefox	IE 9	Opera	Safari
OGG	支持	支持	支持	不支持	不支持
MP3	支持	不支持	支持	不支持	支持
WAV	不支持	支持	不支持	支持	不支持

没有一种通用的文件格式让每个浏览器都使用单个文件格式意味着至少有 2/5 的浏览器无法播放某些声音。这不是无法在单一音频标准中达成一致的浏览器制造商不妥协的问题，而是涉及专利权和特许权使用费的法律和财务问题。不受软件专利限制的 OGG 格式旨在一劳永逸地解决这个问题。然而，在撰写本教材时，Opera 和 Safari 都不支持 OGG。与 OGG 格式的文件相比，可用的 WAV 和 MP3 文件数量要更多，因此毫无疑问，浏览器制造商考虑到了这一点。MP3 作为事实的标准是一个很好的解决方案。

鉴于目前的状况，用户可能认为目前还不是在 HTML5 页面上使用音频的黄金时刻。在某些方面可能的确如此，但是 HTML5 提供了一个解决方案，使用户喜欢的浏览器能够找到一种兼容的格式。

与<audio>标签结合使用时，<source>标签可以嵌套在<audio>容器内。假如想在网页上插入音乐，首先需要获得三种文件类型的音乐，即 OGG、MP3 和 WAV。将这些音乐文件与 HTML5 文件放在同一个文件夹内。然后，将每个文件名放在单独的<source>标签里，并且音频容器中的所有源标签都由<audio></audio>构成，如下代码所示。那么，无论访问者使用什么浏览器，它都将自动选择所读取的第一个文件类型，并播放声音。

```
<audio controls>
<source src="1.ogg" />
<source src="2.mp3" />
<source src="3.wav" />
</audio>
```

一旦用户决定要在网站上提供音频，将面临一个有趣的设计选择。每个浏览器都有与众不同的外观，看起来像是有意识地故意使其与众不同。如图 2.5 所示，展示了这些浏览器控件的外观。

除了 Chrome 浏览器外，所有浏览器都有开始/暂停控件、进度条、滑块、播放秒数、音量/静音控件，还显示声音文件的总秒数。使用 HTML5 标准和浏览器支持，开发人员可以相信用户将拥有与 HTML5 音频类似的体验，因为浏览器控件是类似的。还可以使用 Flash 和 Silverlight 等插件创建控件，但是对于不同的用户，体验可能会有所不同。



图 2.5　各浏览器音频控件外观

某些浏览器(如 IE 9)甚至有自己的声音控制条，在浏览器本身之外运行。用户打开有声音的任何网站时，他们可以从 Windows 任务栏控制声音，并能够预览当前正在播放的声音。

2.8　视　　频

HTML5 提供了 Video 标签，通过 HTML5 的 Video 标签语法，就可以快速地在网页中嵌入影片。但不同浏览器所支持的 HTML5 影片格式(视频解码)皆不同，因此除了要具备

相关的影音文件外，再就是要了解如何让浏览器能自动判断播放的格式，若打开的是 IE 浏览器或 Firefox，它会自动抓取浏览器所支持的格式文件。

Video 标签含有 src、poster、preload、autoplay、loop、controls、width、height 等几个属性，以及一个内部使用的标签<source>。Video 标签内除了可以包含<source>标签外，还可以包含当指定的视频都不能播放时返回的内容。

1. src 属性和 poster 属性

src 属性用于指定视频的地址。而 poster 属性用于指定一张图片，在当前视频数据无效时显示(预览图)。视频数据无效可能是视频正在加载，可能是视频地址错误等。如以下示例代码。

```
<video width="658" height="444" src="1.mp4" poster="1.png"
autoplay="autoplay"></video>
```

2. preload 属性

此属性用于定义视频是否预加载。属性有三个可选择的值：none、metadata、auto。如果不使用此属性，默认为 auto，如下面的代码。

```
<video width="658" height="444" src="1.mp4" poster="1.png"
autoplay="autoplay" preload="none"></video>
```

(1) none：不进行预加载。使用此属性值，可能是页面制作者认为用户不期望此视频，或者减少 HTTP 请求。

(2) metadata：部分预加载。使用此属性值，代表页面制作者认为用户不期望此视频，但为用户提供一些元数据(包括尺寸、第一帧、曲目列表、持续时间等)。

(3) auto：全部预加载。

3. autoplay 属性

autoplay 属性用于设置视频是否自动播放，是一个布尔属性。当出现时，表示自动播放，去掉时表示不自动播放，如下面的代码。

```
<video width="658" height="444" src="1.mp4" poster="1.png"
autoplay="autoplay" preload="none"></video>
```

注意，HTML 中布尔属性的值不是 true 和 false。正确的用法是，在标签中使用此属性表示 true，此时属性要么没有值，要么其值恒等于它的名字(此处，自动播放为<video autoplay/>或者<video autoplay="autoplay"/>)；而在标签中不使用此属性表示 false(此处不进行自动播放为<video/>)。

4. loop 属性

loop 属性用于指定视频是否循环播放，同样是一个布尔属性，如下面的代码。

```
<video width="658" height="444" src="1.mp4" poster="1.png"
autoplay="autoplay" loop="loop"></video>
```

5. controls 属性

controls 属性用于向浏览器指明页面制作者没有使用脚本生成播放控制器，需要浏览器启用本身的播放控制栏。控制栏须包括播放暂停控制，播放进度控制，音量控制等。每个浏览器默认的播放控制栏在界面上不一样，如下面的代码。

```
<video width="658" height="444" src="1.mp4" poster="1.png"
autoplay="autoplay" preload="none" controls="controls"></video>
```

6. width 属性和 height 属性

这两个属于标签的通用属性，用于设置宽度和高度。

7. source 标签

source 标签用于给媒体(因为 audio 标签同样可以包含此标签，所以这里用于媒体，而不是视频)指定多个可选择的(浏览器最终只能选一个)文件地址，且只能在媒体标签没有使用 src 属性时使用。浏览器按 source 标签的顺序检测标签指定的视频是否能够播放(可能是视频格式不支持，视频不存在等)，如果不能播放，换下一个。此方法多用于兼容不同的浏览器。source 标签本身不代表任何含义，不能单独出现，如下面的代码。

```
<video width="658" height="444" poster="1.png" autoplay="autoplay"
 preload="none" controls="controls"><source src="1.ogv" /><source
src="1.ogg" /></video>
```

此标签包含 src、type、media 三个属性。

(1) src 属性：用于指定媒体的地址，和 video 标签的一样。

(2) type 属性：用于说明 src 属性指定媒体的类型，帮助浏览器在获取媒体前判断是否支持此类别的媒体格式。

(3) media 属性：用于说明媒体在何种媒介中使用，不设置时默认值为 all，表示支持所有媒介。

2.9 SVG

可缩放矢量图形(Scalable Vector Graphics，SVG)是基于矢量的图形家族的一部分。它们与基于光栅的图形不同，后者在一个数据数组中存储每个像素的颜色定义。如今网络上使用的最常见的光栅图形格式包括 JPEG、GIF 和 PNG，每种格式都具有优缺点。其他图片格式缩写词有：GIF(图形交换格式)、JPEG(联合图像专家组)、PNG(可移植网络图形)、SVG(可缩放矢量图形)。相比任何基于光栅的格式，SVG 具有多项优势。

SVG 图形是使用数学公式创建的，需要在源文件中存储的数据要少得多，因为无须存储每个独立像素的数据。

矢量图形可更好地缩放。对于网络上的图像，尝试从原始大小放大图像可能产生失真(或像素化的)图像。原始像素数据是针对特定大小进行设计的。当图像不再是该大小时，显示图像的程序会猜测使用何种数据来填充新的像素。矢量图像具有更高的弹性；当图像大小变化时，数据公式可相应地调整。

源文件大小可能更小，所以 SVG 图形比其他光栅图形的加载速度更快，使用的带宽更少。

SVG 图像由浏览器渲染，可以以编程方式绘制。SVG 图像可动态地更改，这使它们尤其适合数据驱动的应用程序，比如图表。

SVG 图像的源文件是一个文本文件，所以它具有易于访问和搜索引擎友好的特征。

本节将介绍 SVG 格式的优势，以及它们如何在 HTML5 中的 Web 设计工作中提供帮助。

2.9.1 SVG 基础

要创建 SVG 图形，就会经历与创建 JPEG、GIF 或 PNG 文件完全不同的流程。JPEG、GIF 和 PNG 文件通常使用图像编辑程序创建，比如 Adobe Photoshop。SVG 图像通常使用基于 XML 的语言创建。有一些 SVG 编辑 GUI 将为用户生成基础的 XML。但是，对于本文，假设用户使用的是原始的 XML。请参见参考资料获取有关 SVG 编辑程序的信息。

以下代码给出了一个简单 SVG XML 文件的示例，该文件绘制一个具有 2px 宽的黑色边框的红色圆形。显示效果如图 2.6 所示。

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
 <circle  cx="100" cy="50" r="40" stroke="black" stroke-width="2" fill="red" />
</svg>
```

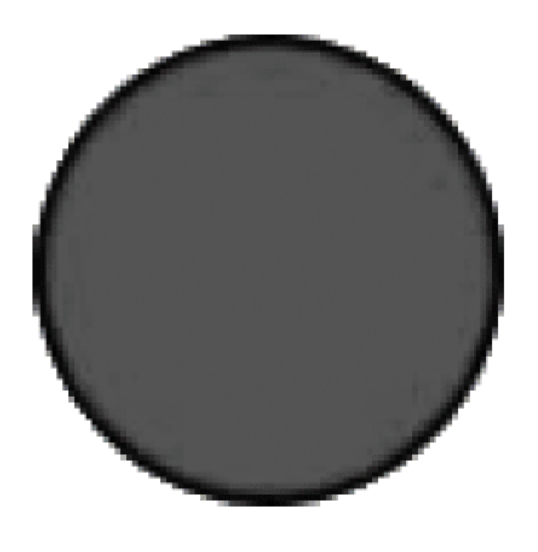


图 2.6　简单 SVG XML 文件的示例

2.9.2 创建基本形状

对于 SVG 图形，需要使用 XML 标记来创建形状。表 2.4 给出了这些 XML 元素。

表 2.4　XML 元素

元　素	描　述	元　素	描　述
line	创建一条简单的线	ellipse	创建一个椭圆
polyline	定义由多个线定义构成的形状	polygon	创建一个多边形
rect	创建一个矩形	path	支持任意路径的定义
circle	创建一个圆形		

1. line 元素

line 元素是基本的绘图元素。以下代码展示了如何创建一条水平线，显示效果如图 2.7 所示。

```
<svg xmlns = "http://www.w3.org/2000/svg" version = "1.1"
    width = "100 % " height = "100 % " >
    <line x1 = "25" y1 = "150" x2 = "300" y2 = "150"
      style = "stroke:red;stroke - width:10"/>
</svg>
```

图 2.7　水平线

根 SVG 标记具有宽度和高度属性,用于定义可用于绘制的画布区域。它们的原理类似于其他 HTML 元素的宽度和高度属性。在本例中,画布设置为占据所有可用空间。

该示例还使用了 style 标记。SVG 图形支持使用多种方法设置其内容的样式。本文中的样式可用于使它们变得显眼,也可在必须使用某些属性(比如笔画颜色和宽度)才能渲染图像时使用。

要创建一个线定义,可以定义相对于画布的开始和结束 x 和 y 坐标。x1 和 y1 属性是开始坐标,x2 和 y2 属性是结束坐标。要更改线的方向,只需更改这些坐标。例如,通过修改上一个示例,可以生成一条对角线,如以下代码所示,显示效果如图 2.8 所示。

```
<svg xmlns = "http://www.w3.org/2000/svg" version = "1.1"
    width = "100 % " height = "100 % " >
  <line x1 = "0" y1 = "0" x2 = "200" y2 = "200" style = "stroke:red;stroke - width:10"/>
</svg>
```

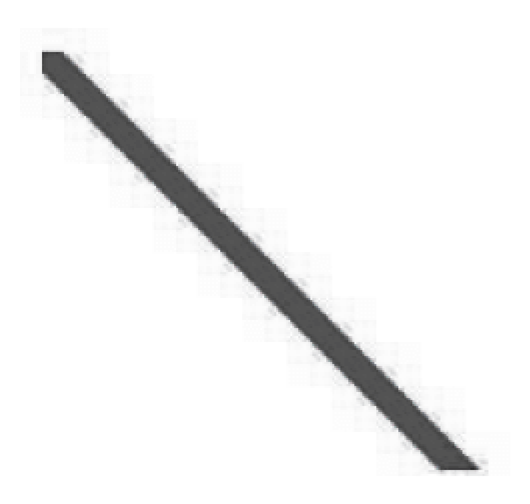


图 2.8　对角线

2. polyline 元素

多直线图形是一个由多个线定义组成的图形。以下的代码中创建了一个类似一组楼梯的图形,显示效果如图 2.9 所示。

```
<svg xmlns = http://www.w3.org/2000/svg width = "100 % " height = "100 % " version = "1.1">
    <polyline points = "0,40 40,40 40,80 80,80 80,120 120,120 120,160"
     style = "fill:white;stroke:red;stroke - width:4"/>
</svg>
```

要创建一个多直线图形,可以使用 points 属性并定义由逗号分隔的 x 和 y 坐标对。在本例中,第一个点定义为 0、40,其中 0 是 x 值,40 是 y 值。但是,单独一组点无法在屏幕上显示任何东西,因为这仅告诉 SVG 渲染器从何处开始。在最低限度上,需要两组点:一个

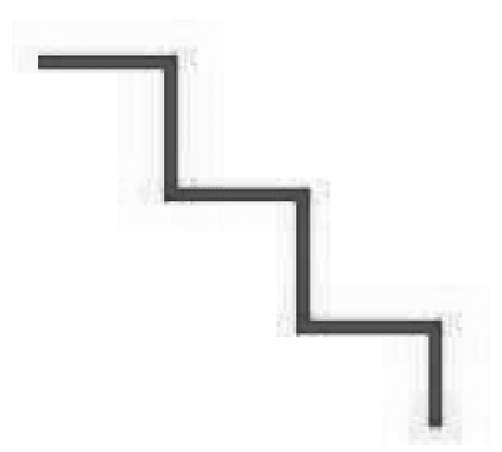


图 2.9　多直线

开始点和一个结束点(例如 points="0,40 40,40”)。

与简单的线图形一样,这些线不需要完全水平或垂直。如果更改上一个示例中的值,如以下的代码所示,可以生成不规则的线形状,如图 2.10 所示。

```
<svg xmlns = "http://www.w3.org/2000/svg" width = "100 % " height = "100 % "
version = "1.1">
   <polyline points = "20,20 40,25 60,40 80,120 120,140 200,180"
         style = "fill:white;stroke:red;stroke - width:3"/>
</svg>
```



图 2.10　不规则形状

3. rect 元素

创建一个矩形非常简单,只需定义宽度和高度,如以下代码所示,显示效果如图 2.11 所示。

```
<svg xmlns = http://www.w3.org/2000/svg width = "100 % " height = "100 % "
     version = "1.1">
  <rect width = "300" height = "100" style = "fill:red"/>
</svg>
```



图 2.11　矩形

同样,也可以使用 rect 标记创建一个正方形,正方形就是高和宽相等的矩形。

4. circle 元素

要创建一个圆,可以定义圆心的 x 和 y 坐标和一个半径,如以下代码所示,显示效果如图 2.12 所示。

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
      <circle cx="100" cy="50" r="40" fill="red"/>
</svg>
```

cx 和 cy 属性定义圆心相对于绘图画布的位置。因为半径是圆宽度的一半，所以在定义半径时，请记住图像的总宽度将是该值的两倍。

5. ellipse 元素

椭圆基本上是一个圆，其中的代码定义了两个半径，如以下代码所示，显示效果如图 2.13 所示。

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
      <ellipse cx="300" cy="80" rx="100" ry="50" style="fill:red;"/>
</svg>
```

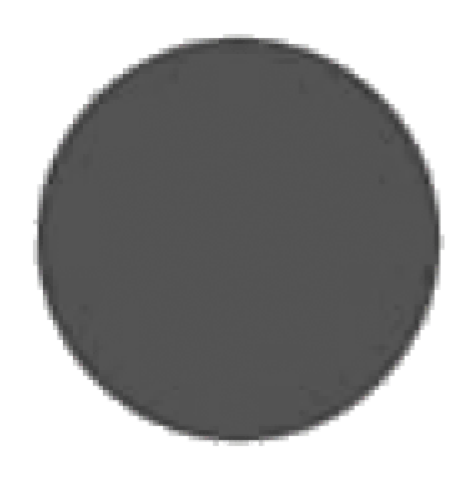


图 2.12 圆

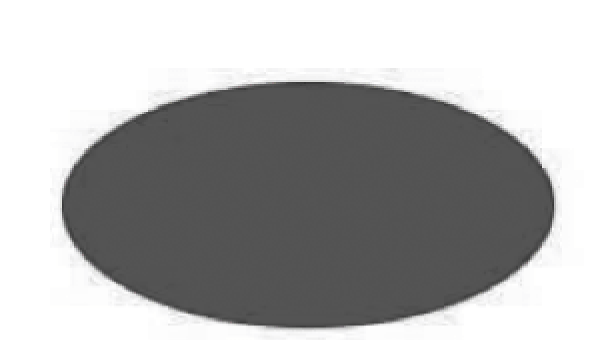


图 2.13 椭圆

再次说明，cx 和 cy 属性定义了相对于画布的中心坐标。但是对于椭圆，需要使用 rx 和 ry 属性为 x 轴定义一个半径，为 y 轴定义一个半径。

6. polygon 元素

多边形这个形状包含至少三条边。以下代码创建了一个简单的三角形，显示效果如图 2.14 所示。

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
       <polygon points="200,10 250,190 160,210"
            style="fill:red;stroke:black;stroke-width:1"/>
</svg>
```

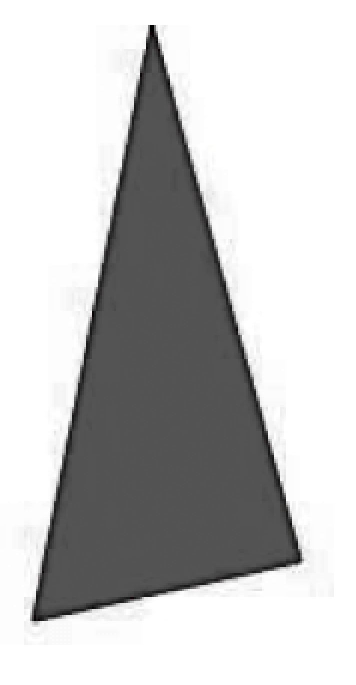


图 2.14 多边形

类似于 polyline 元素，可使用 points 属性定义几对 x 和 y 坐标来创建多边形。可以通过添加 x 和 y 对，创建具有任意多条边的多边形。通过修改上一个示例，可以创建一个四边形，如以下代码所示，显示效果如图 2.15 所示。

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
     <polygon points="220,10 300,210 170,250 123,234"
     style="fill:red;stroke:black;stroke-width:1"/>
</svg>
```

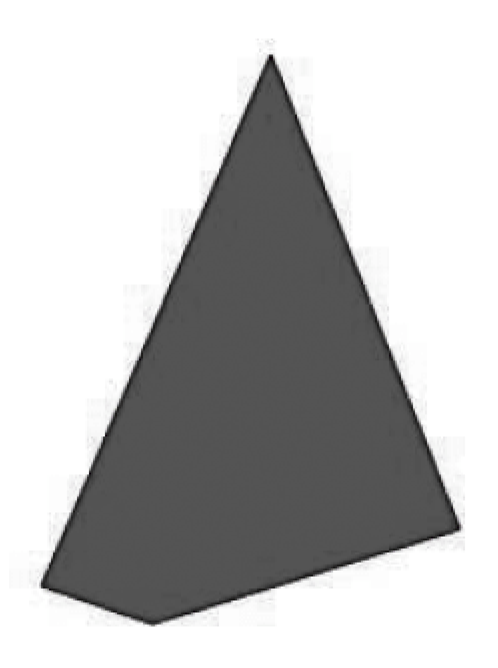


图 2.15　四边形

甚至可以使用 polygon 标记创建复杂的形状。如以下代码所示，显示效果如图 2.16 所示。

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
      <polygon points="100,10 40,180 190,60 10,60 160,180 100,10"
      style="fill:red;stroke:black;stroke-width:1"/>
</svg>
```



图 2.16　五角星

7. path 元素

使用 path 元素(所有绘图元素中最复杂的)，可以使用一组专门的命令创建任意图形。path 元素支持表 2.5 中的命令。

能以大写或小写形式使用这些命令。当命令为大写时，应用绝对位置。当它为小写时，应用相对位置。提供所有命令示例已超出了本文的范围。但是，以下示例会演示它们的基本使用。

表 2.5　path 元素支持的命令

命　令	描　　述	命　令	描　　述
M	移动到	S	使用平滑曲线连接到
L	连线到	Q	使用二次贝塞尔曲线连接到
H	水平连线到	T	使用平滑的二次贝塞尔曲线连接到
V	垂直连线到	A	使用椭圆曲线连接到
C	使用曲线连接到	Z	将路径封闭到

可以使用 path 元素从本文前面的示例创建任何简单的形状。以下代码使用 path 元素创建了一个基本的三角形，显示效果如图 2.17 所示。

```
<svg xmlns = "http://www.w3.org/2000/svg" version = "1.1">
  <path d = "M150 0 L75 200 L225 200 Z" style = "fill:red"/>
</svg>
```

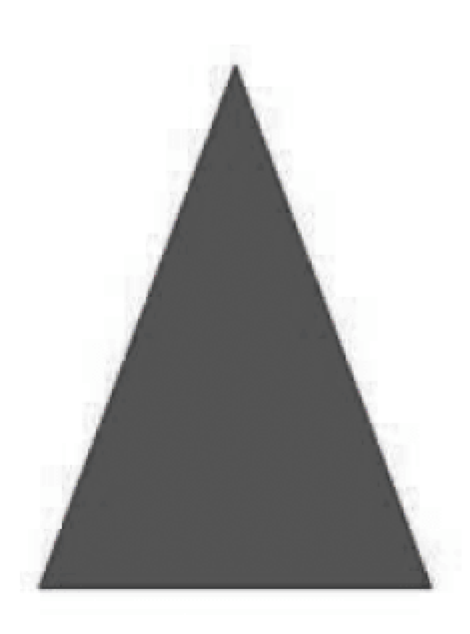


图 2.17　三角形

这组命令使用 d 属性定义。在本例中，从 x 坐标 150 和 y 坐标 0 处开始绘制，这在移动到命令(M150 0)中定义。然后再使用“连线到”命令绘制一条直线连接到 x 坐标 75 和 y 坐标 200 的位置(L75 200)。接下来，使用另一个“连线到”命令绘制另一条线(L225 200)。最后，使用“封闭到”命令封闭图形(Z)。Z 命令没有提供任何坐标，因为要关闭所在的路径，根据定义，要绘制一条从当前位置到图形起点(在本例中为 x=150 y=0)的线。

这里的意图是展示一个基本示例，如果想绘制的只是一个简单的三角形，最好使用 polygon 标记。

path 元素的真正强大之处是创建自定义形状的能力，如以下代码所示，显示效果如图 2.18 所示。

```
<svg xmlns = "http://www.w3.org/2000/svg" version = "1.1">
     <path d = "M300,200 h - 150 a150,150 0 1,0 150, - 150 z"
               fill = "red" stroke = "blue" stroke - width = "5"/>
     <path d = "M275,175 v - 150 a150,150 0 0,0  - 150,150 z"
               fill = "yellow" stroke = "blue" stroke - width = "5"/>
     <path d = "M600,350 l 50, - 25
               a25,25  - 30 0,1 50, - 25 l 50, - 25
               a25,50  - 30 0,1 50, - 25 l 50, - 25
```

```
                a25,75 -30 0,1 50,-25 l 50,-25
                a25,100 -30 0,1 50,-25 l 50,-25"
                fill="none" stroke="red" stroke-width="5"/>
</svg>
```

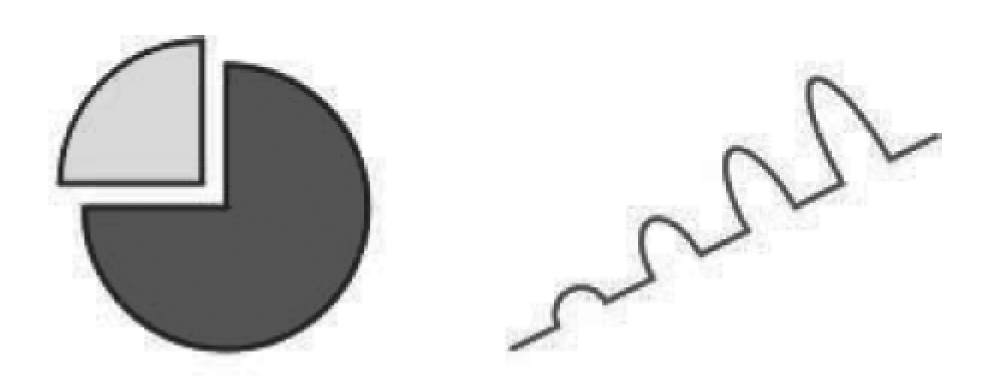


图 2.18　创建自定义形状

使用 path 元素，可以创建复杂的图形，比如图表和波浪线。请注意，这个示例使用了多个 path 元素。当创建图形时，根 SVG 标记中可以包含多个绘图元素。

2.9.3　过滤器和渐变

除了目前为止许多示例中的基本 CSS 样式，SVG 图形还支持使用过滤器和渐变。本节将介绍如何向 SVG 图形应用过滤器和渐变。

可以使用过滤器向 SVG 图形应用特殊的效果。SVG 支持以下过滤器：feBlend、feColorMatrix、feComponentTransfer、feComposite、feConvolveMatrix、feDiffuseLighting、feDisplacementMap、feFlood、feGaussianBlur、feImage、feMerge、feMorphology、feOffset、feSpecularLighting、feTile、feTurbulence、feDistantLight、fePointLight、feSpotLight 等。

以下代码创建了一种应用到矩形上的投影效果，显示效果如图 2.19 所示。

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
   <defs>
   <filter id="f1" x="0" y="0"  width="200%" height="200%">
       <feOffset result="offOut" in="SourceAlpha" dx="20" dy="20"/>
       <feGaussianBlur result="blurOut" in="offOut" stdDeviation="10"/>
       <feBlend in="SourceGraphic" in2="blurOut" mode="normal"/>
   </filter>
   </defs>
    <rect width="90" height="90" stroke="green"
          stroke-width="3" fill="yellow" filter="url(#f1)"/>
</svg>
```

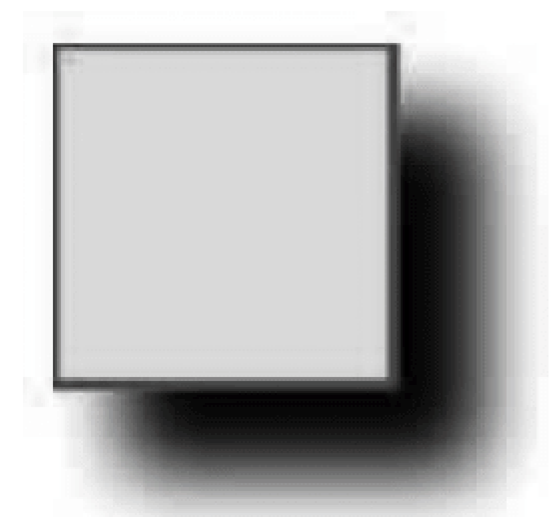


图 2.19　矩形的投影效果

过滤器在 defs(表示定义)元素中定义。本示例中的过滤器分配了一个 id 为“f1”。filter 标记本身拥有定义过滤器的 x 和 y 坐标及宽度和高度的属性。在 filter 标记中,可以使用想要的过滤器元素并将其属性设置为想要的值。

定义过滤器之后,使用 filter 属性将它与一个特定图形关联,如 rect 元素中所示。将 url 值设置为用户分配给过滤器的 id 属性的值。

渐变是从一种颜色到另一种颜色逐渐的过渡。渐变具有两种基本形式:线性和径向渐变。所应用的渐变类型由用户使用的元素确定。以下示例展示了应用于一个椭圆形的线性和径向渐变。显示效果如图 2.20 和图 2.21 所示。

```
<svg xmlns = "http://www.w3.org/2000/svg" version = "1.1">
  <defs>
  <linearGradient id = "grad1" x1 = "0%" y1 = "0%" x2 = "100%" y2 = "0%">
    <stop offset = "0%" style = "stop-color:rgb(255,255,0);stop-opacity:1"/>
    <stop offset = "100%" style = "stop-color:rgb(255,0,0);stop-opacity:1"/>
   </linearGradient>
  </defs>
  <ellipse cx = "200" cy = "70" rx = "85" ry = "55" fill = "url(#grad1)"/>
</svg>
```

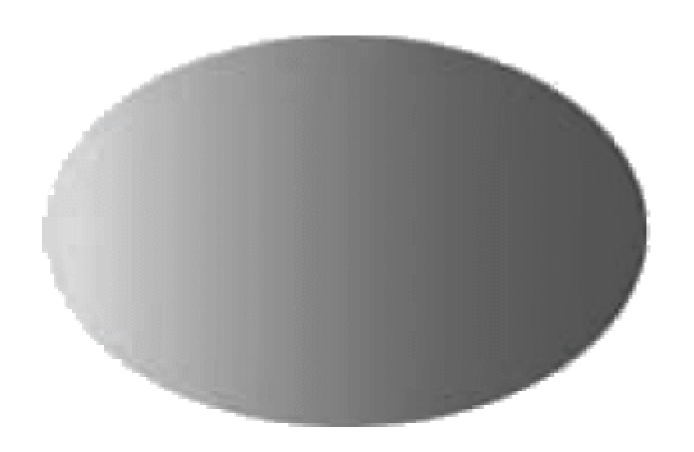


图 2.20　线性渐变椭圆

```
<svg xmlns = "http://www.w3.org/2000/svg" version = "1.1">
  <defs>
  <radialGradient id = "grad1" cx = "50%" cy = "50%"  r = "50%" fx = "50%" fy = "50%">
  <stop offset = "0%" style = "stop-color:rgb(255,255,255);stop-opacity:0"/>
  <stop offset = "100%" style = "stop-color:rgb(255,0,0);stop-opacity:1"/>
  </radialGradient>
  </defs>
  <ellipse cx = "200" cy = "70" rx = "85" ry = "55" fill = "url(#grad1)"/>
</svg>
```

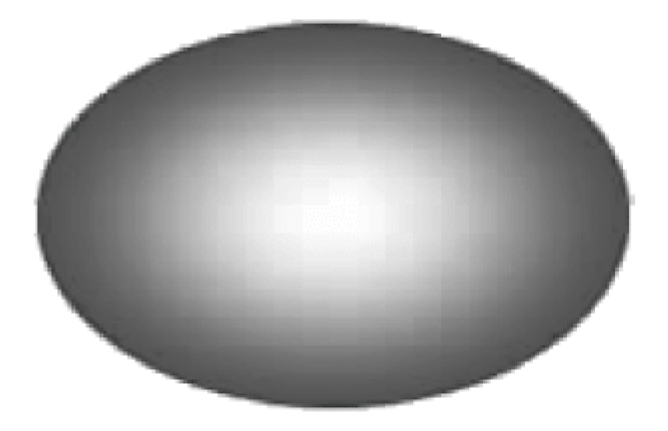


图 2.21　径向渐变椭圆

像过滤器一样，渐变在 defs 元素内定义。每个渐变分配有一个 id。渐变属性(比如颜色)可使用 stop 元素在渐变标记内设置。要将渐变应用于图形，可以将 fill 属性的 url 值设置为想要的渐变的 id。

2.9.4 SVG 与文本

除了基本形状，还可以使用 SVG 生成文本，如以下代码所示，显示效果如图 2.22 所示。

```
<svg xmlns = "http://www.w3.org/2000/svg" version = "1.1">
    <text x = "0" y = "15" fill = "red"> I love SVG</text>
</svg>
```

I love SVG

图 2.22　SVG 文本

此示例使用了一个 text 元素来创建句子"I love SVG"。当使用 text 元素时，要显示的实际文本在开始和结束 text 元素之间。

可以沿多个轴，以及甚至沿多条路径显示文本。以下代码沿一条弧形路径显示文本，显示效果如图 2.23 所示。

```
<svg xmlns = "http://www.w3.org/2000/svg" version = "1.1"
      xmlns:xlink = "http://www.w3.org/1999/xlink">
      <defs>
          <path id = "path1" d = "M75,20 a1,1 0 0,0 100,0"/>
      </defs>
      <text x = "10" y = "100" style = "fill:red;">
          <textPath xlink:href = "#path1"> I love SVG I love SVG</textPath>
      </text>
</svg>
```



图 2.23　沿弧形路径显示文本

在此示例中，向根 SVG 标记添加了一个额外的 XML 命名空间 xlink。用户显示文本的弧形路径在 defs 元素内创建，所以该路径不会在图形中实际渲染出来。要显示的文本嵌套在一个 textPath 元素内，该元素使用 xlink 命名空间引用想要的路径的 id。

与其他 SVG 图形一样，也可以向文本应用过滤器和渐变。以下代码向一些文本应用了一个过滤器和一种渐变，显示效果如图 2.24 所示。

```
<svg xmlns = "http://www.w3.org/2000/svg" version = "1.1"
      xmlns:xlink = "http://www.w3.org/1999/xlink">
```

```
    <defs>
<radialGradient id="grad1" cx="50%" cy="50%" r="50%" fx="50%" fy="50%">
    <stop offset="0%" style="stop-color:red; stop-opacity:0"/>
    <stop offset="100%" style="stop-color:rgb(0,0,0);stop-opacity:1"/>
</radialGradient>
<filter id="f1" x="0" y="0" width="200%" height="200%">
     <feOffset result="offOut" in="SourceAlpha" dx="20" dy="20"/>
     <feGaussianBlur result="blurOut" in="offOut" stdDeviation="10"/>
     <feBlend in="SourceGraphic" in2="blurOut" mode="normal"/>
  </filter>
 </defs>
 <text x="10" y="100" style="fill:url(#grad1); font-size: 30px;"
    filter="url(#f1)">I love SVG I love SVG
 </text>
</svg>
```

I love SVG I love SVG

图 2.24　向文本应用过滤器和渐变

2.9.5　向网页添加 SVG XML

创建 SVG XML 之后，可采用多种方式将它包含在 HTML 页面中。第一种方法是直接将 SVG XML 嵌入到 HTML 文档中，如以下代码所示。

```
<html>
    <head>
    <title>Embedded SVG</title    >
    </head>
    <body style="height: 600px;width: 100%; padding: 30px;">
      <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
        <circle cx="100" cy="50" r="40" fill="red"/>
        </svg>
    </body>
</html>
```

此方法可能最简单，但它不支持重用。请记住，可以使用 SVG 扩展名保存 SVG XML 文件。当将 SVG 图形保存到 SVG 文件中时，可以使用 embed、object 和 iframe 元素来将它包含在网页中。如以下代码显示了使用 embed 元素包含 SVG XML 文件。

```
<embed src="circle.svg" type="image/svg+xml" />
```

如以下代码显示了如何使用 object 元素包含一个 SVG XML 文件。

```
<object data="circle.svg" type="image/svg+xml"></object>
```

如以下代码给出了使用 iframe 元素包含一个 SVG XML 文件。

```
<iframe src="circle1.svg"></iframe>
```

当使用其中一种方法时，可以将同一个 SVG 图形包含在多个页面中，并编辑 SVG 源文件来进行更新。

2.10 拖　　放

拖放是一种常见的特性，即抓取对象以后拖到另一个位置。在 HTML5 中，拖放是标准的一部分，任何元素都能够拖放。下面的例子是一个简单的拖放实例。

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
function allowDrop(ev)
{
ev.preventDefault();
}

function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}

function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)"
ondragover="allowDrop(event)"></div>
<img id="drag1" src="img_logo.gif" draggable="true"
ondragstart="drag(event)" width="336" height="69" />

</body>
</html>
```

对于上面的代码解释如下：首先，为了使元素可拖动，把 draggable 属性设置为 true。；拖动什么用 ondragstart 和 setData()。然后，规定当元素被拖动时，会发生什么。在上面的例子中，ondragstart 属性调用了一个函数 drag(event)，它规定了被

拖动的数据。dataTransfer.setData()方法设置被拖数据的数据类型和值。

```
function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}
```

在这个例子中，数据类型是 Text，值是可拖动元素的 id("drag1")。放到何处用 ondragover。ondragover 事件规定在何处放置被拖动的数据。默认地，无法将数据/元素放置到其他元素中。如果需要设置允许放置，必须阻止对元素的默认处理方式。这要通过调用 ondragover 事件的 event.preventDefault()方法，放置时用 ondrop，当放置被拖数据时，会发生 drop 事件。在上面的例子中，ondrop 属性调用了一个函数 drop(event)。

```
function drop(ev)
{
ev.preventDefault();
var data = ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
```

调用 preventDefault()来避免浏览器对数据的默认处理(drop 事件的默认行为是以链接形式打开的)，通过 dataTransfer.getData("Text")方法获得被拖的数据。该方法将返回在 setData()方法中设置为相同类型的任何数据。被拖数据是被拖元素的 id("drag1")把被拖元素追加到放置元素(目标元素)中。

2.11 HTML5 移动开发实例

下面来看一个使用 Canvas 来创建一个图形报告的应用程序。图 2.25 展示了该应用程序的屏幕截图：一个显示每年结果的柱状图。代码如下。

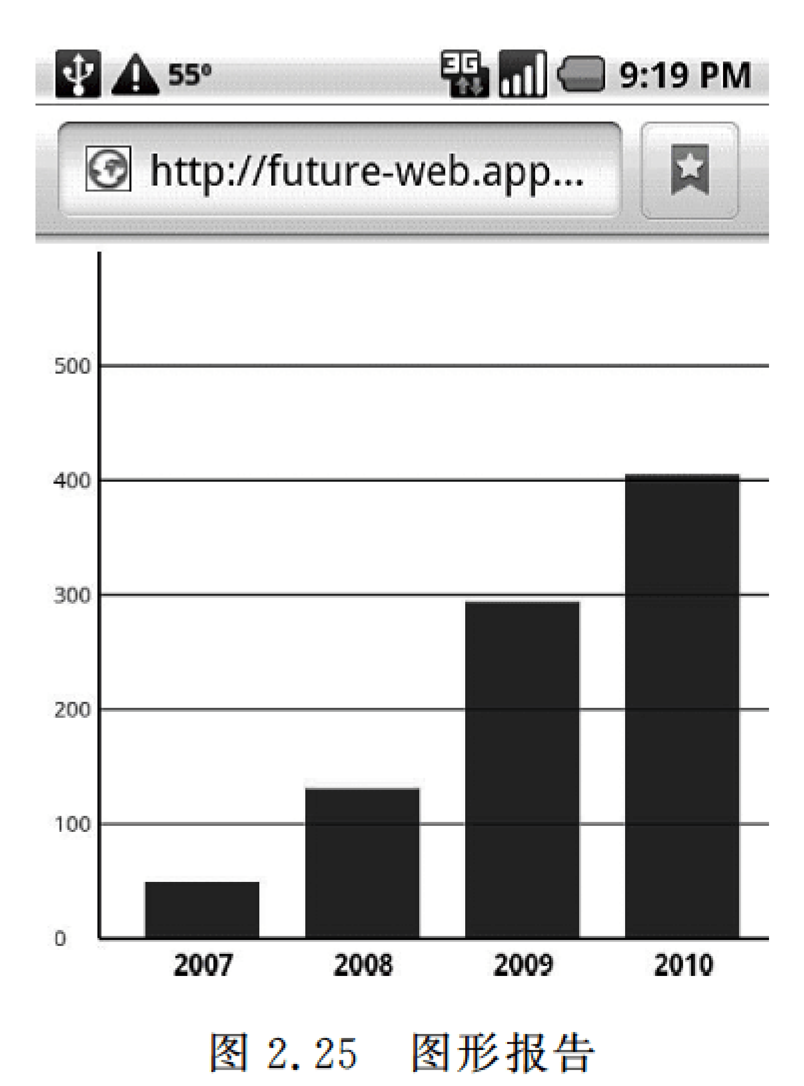


图 2.25 图形报告

HTML 清单如下。

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width; initial-scale=1.0;
        maximum-scale=1.0; user-scalable=0;"/>
    <meta name="apple-touch-fullscreen" content="YES" />
    <title>HTML 5 Reports</title>
    <script type="text/javascript">
        function init(){
            var data = [{year : "2007",sales : 49},
                {year : "2008",sales : 131},
                {year : "2009",sales : 294},
                {year : "2010",sales : 405}];
            var report = {x : "year",
                    y : "sales",
                    values : data};
            graph(report, 350, 300);
        }
    </script>
</head>
<body onload="init()">
    <canvas id="graph"></canvas>
</body>
</html>
```

Graph 函数如下。

```
function graph(report, maxWidth, maxHeight){
    var data = report.values;
    var canvas = document.getElementById("graph");
    var axisBuffer = 20;
    canvas.height = maxHeight + 100;
    canvas.width = maxWidth;
    var ctx = canvas.getContext("2d");

    var width = 50;
    var buffer = 20;
    var i = 0;
    var x = buffer + axisBuffer;
    ctx.font = "bold 12px sans-serif";
    ctx.textAlign = "start";
    for (i=0;i<data.length;i++){
        ctx.fillStyle = "rgba(0, 0, 200, 0.9)";
        ctx.fillRect(x, maxHeight - (data[i][report.y] / 2),
                 width, (data[i][report.y] / 2));
        ctx.fillStyle = "rgba(0, 0, 0, 0.9)";
```

```
            ctx.fillText(data[i][report.x], x + (width / 4), maxHeight + 15);
            x += width + buffer;
        }

        //draw the horizontal axis
        ctx.moveTo(axisBuffer, maxHeight);
        ctx.lineTo(axisBuffer + maxWidth, maxHeight);
        ctx.strokeStyle = "black";
        ctx.stroke();

        //draw the vertical axis
        ctx.moveTo(axisBuffer,0);
        ctx.lineTo(axisBuffer,maxHeight);
        ctx.stroke();

        //draw gridlines
        var lineSpacing = 50;
        var numLines = maxHeight/lineSpacing;
        var y = lineSpacing;
        ctx.font = "10px sans-serif";
        ctx.textBaseline = "middle";
        for (i = 0;i < numLines;i++){
            ctx.strokeStyle = "rgba(0,0,0,0.25)";
            ctx.moveTo(axisBuffer, y);
            ctx.lineTo(axisBuffer + maxWidth,y);
            ctx.stroke();
            ctx.fillStyle = "rgba(0,0,0, 0.75)";
            ctx.fillText("" + (2 * (maxHeight - y)), 0, y);
            y += lineSpacing;
        }
    }
```

小　　结

本章主要介绍了 HTML5 中新增的语义化标签元素，以及结合移动设备环境下的 Web 页面布局以及相关的例子。通过学习 HTML5，可以为以后从事移动 Web 开发工作打下基础。

第 3 章 HTML5 本地存储

本章学习目标

- localStorage 数据存储
- sessionStorage 数据存储
- Web SQL Database 数据存储

在 HTML5 中引入了 localStorage，其实这个概念很早就有，如最初的 Cookies，但是 cookie 不适合大量数据的存储，因为它们由每个对服务器的请求来传递，这使得 cookie 的速度很慢而且效率也不高。后来又有 IE 的 userData，再到 Google 的 Gears，直到现在的 localStorage，从概念上都是一脉相承的。只不过在 HTML5 中将它制定成了标准，结束了以前各行其是的状态。HTML5 提供了两种在客户端存储数据的新方法：localStorage(没有时间限制的数据存储)和 sessionStorage(针对一个 session 的数据存储)。

sessionStorage 用于本地存储一个会话(Session)中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此 sessionStorage 不是一种持久化的本地存储，仅仅是会话级别的存储。而 localStorage 用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

3.1 HTML5 本地存储的浏览器支持情况

除了 IE 7 及以下不支持 HTML5 外，其他标准浏览器都完全支持(IE 及 Firefox 需在 Web 服务器里运行)。值得一提的是，IE 总是办好事，例如 IE 7、IE 6 中的 UserData 其实就是 JavaScript 本地存储的解决方案。通过简单的代码封装可以统一到所有的浏览器都支持 Web 存储。

要判断浏览器是否支持 localStorage 可以使用下面的代码。

```
if(window.localStorage){
  alert("浏览支持 localStorage")
 }else{
alert("浏览暂不支持 localStorage") }
//或者
if(typeof window.localStorage == 'undefined'){
    alert("浏览暂不支持 localStorage")
}
```

3.2 sessionStorage 操作

在 HTML5 中增加了一个 JavaScript 对象：sessionStorage；通过此对象可以直接操作存储在浏览器中的会话级别的 Web 存储。存储在 sessionStorage 中的数据首先是 Key-Value 形式的，另外就是它跟浏览器当前会话相关，当会话结束后，数据会自动清除，跟未设置过期时间的 Cookie 类似。sessionStorage 提供了 4 个方法对本地存储进行相关操作。

(1) setItem(key,value)：添加本地存储数据。

(2) getItem(key)：通过 key 获取相应的 value。

(3) removeItem(key)：通过 key 删除本地数据。

(4) clear()：清空数据。

示例代码如下。

```
<script type="text/javascript">
//添加 key-value 数据到 sessionStorage
sessionStorage.setItem("demokey", "abc");
//通过 key 来获取 value
var dt = sessionStorage.getItem("demokey");
alert(dt);
//清空所有的 key-value 数据
//sessionStorage.clear();
alert(sessionStorage.length);
</script>
```

对于 JavaScript 的学习和调试必须有 Chrome 的调试工具辅助才能事半功倍，使用 F12 快捷键就立即打开工具了，IE 也是这个快捷键。如图 3.1 所示，可以查看当前浏览器中的 sessionStorage 数据。

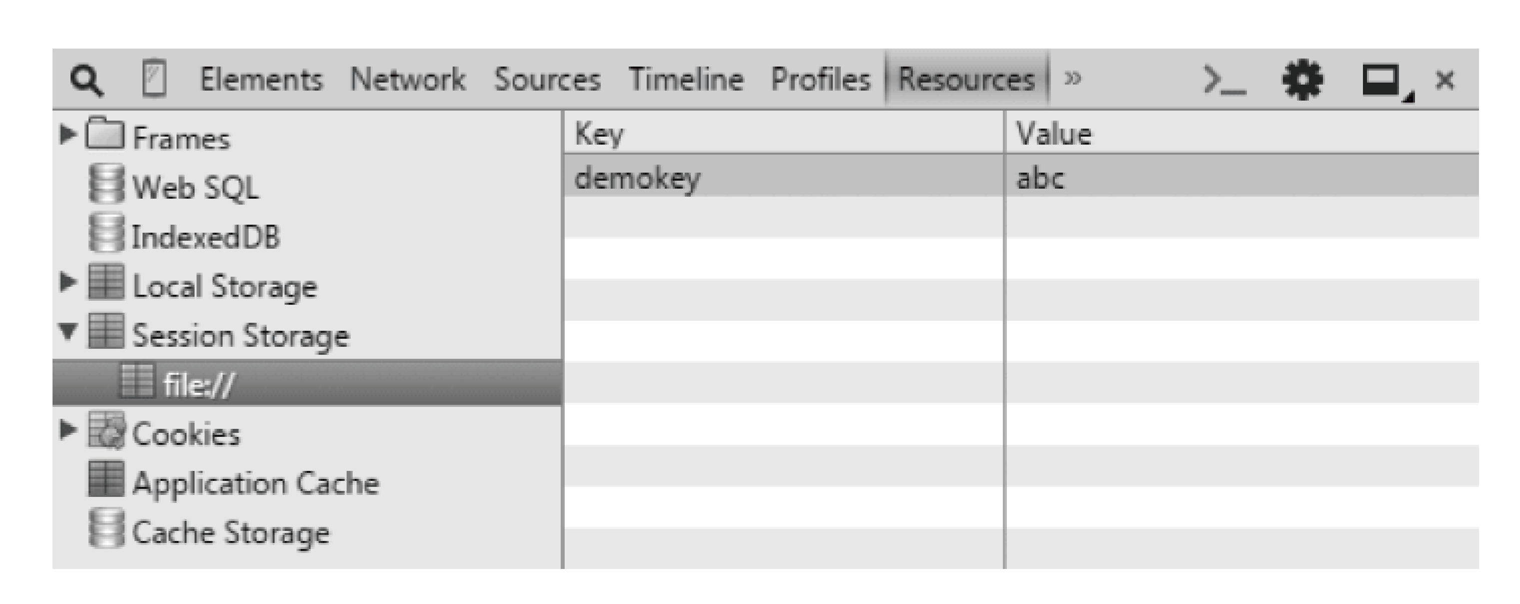


图 3.1 浏览器中的 sessionStorage 数据

3.3 localStorage 操作

在最新的 JavaScript 的 API 中增加了 localStorage 对象，以便于用户存储永久存储的 Web 端的数据。而且数据不会随着 HTTP 请求发送到后台服务器，而且存储数据的大小机会不用考虑，因为在 HTML5 的标准中要求浏览器至少要支持到 4MB。所以，这完全颠

覆了 Cookie 的限制，为 Web 应用在本地存储复杂的用户痕迹数据提供了非常方便的技术支持。接下来分别介绍一下 localStorage 的常用方法，当然基本上与 sessionStorage 是一致的。localStorage 提供了 4 个方法对本地存储进行相关操作。

(1) setItem(key,value)：添加本地存储数据。

(2) getItem(key)：通过 key 获取相应的 value。

(3) removeItem(key)：通过 key 删除本地数据。

(4) clear()：清空数据。

示例代码如下。

```
<script type = "text/javascript">
    //添加 key - value 数据到 sessionStorage
    localStorage.setItem("demokey", "abc");
    //通过 key 来获取 value
    var dt = localStorage.getItem("demokey");
    alert(dt);
    //清空所有的 key - value 数据
    //localStorage.clear();
    alert(localStorage.length);
</script>
```

同样用 Chrome 的调试工具可以查看当前浏览器中的 localStorage 数据，如图 3.2 所示。

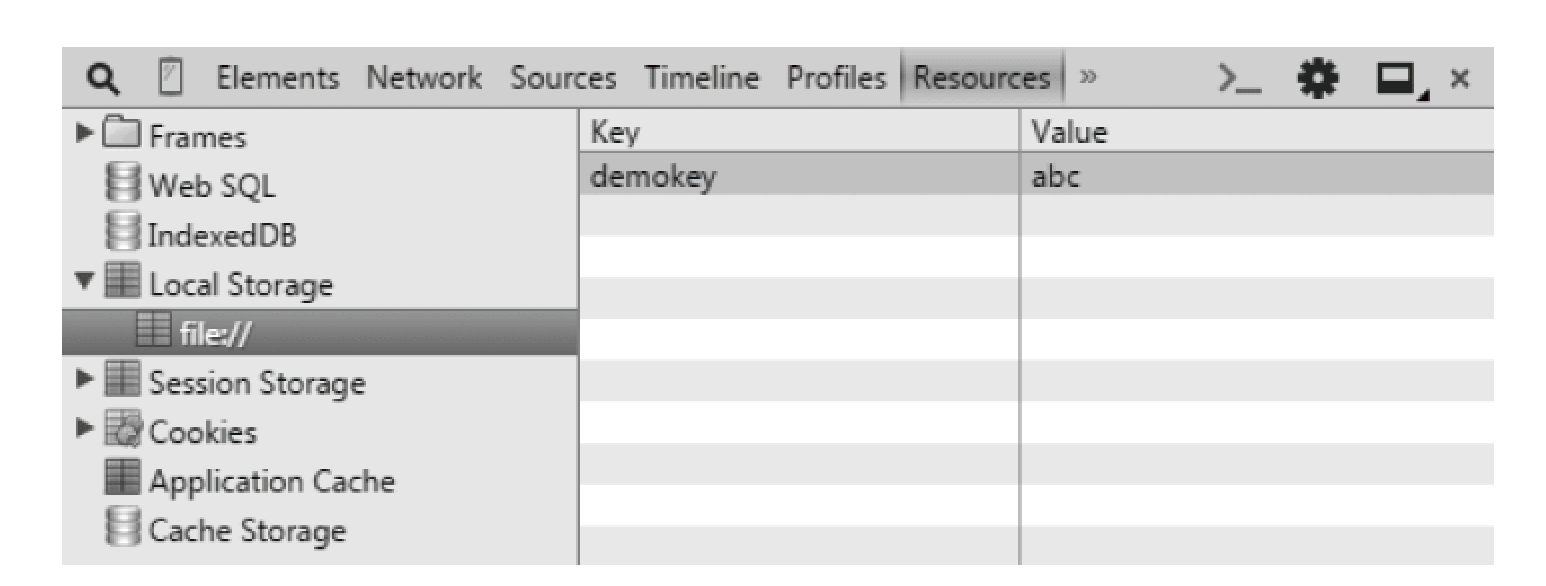


图 3.2 浏览器中的 localStorage 数据

3.4 Web SQL Database

虽然 HTML5 已经提供了功能强大的 localStorage 和 sessionStorage，但是它们两个都只能提供存储简单数据结构的数据，对于复杂的 Web 应用的数据却无能为力。HTML5 还提供了一个浏览器端的数据库支持(Web SQL Database)，它允许直接通过 JavaScript 的 API 在浏览器端创建一个本地的数据库，而且支持标准的 SQL 的 CRUD 操作，让离线的 Web 应用更加方便地存储结构化的数据。接下来介绍本地数据的相关 API 和用法。

操作本地数据库的最基本的步骤如下。

第一步：使用 openDatabase 方法创建一个访问数据库的对象。

第二步：使用第一步创建的数据库访问对象来执行 transaction 方法，通过此方法可以

设置一个开启事务成功的事件响应方法，在事件响应方法中可以执行 SQL。

第三步：通过 executeSql 方法执行查询，当然查询可以是 CRUD 等操作。

接下来分别介绍一下相关方法的参数和用法。

（1） openDatabase 方法。

例如，获取或者创建一个数据库，如果数据库不存在那么创建之。例如：

```
var dataBase = openDatabase("student","1.0","学生表",1024 * 1024,function () { });
```

openDatabase 方法打开一个已经存在的数据库，如果数据库不存在，它还可以创建数据库。几个参数的意义分别如下。

① 数据库名称。

② 数据库的版本号，目前来说使用 1.0 就可以了，当然也可以不填。

③ 对数据库的描述。

④ 设置分配的数据库的大小（单位是 KB）。

⑤ 回调函数（可省略）。

初次调用时创建数据库，以后就是建立连接了。

（2） db.transaction 方法可以设置一个回调函数，此方法用于处理事务，当一条语句执行失败的时候，整个事务回滚。该方法有以下三个参数。

① 包含事务内容的一个方法。

② 执行成功回调函数（可选）。

③ 执行失败回调函数（可选）。

（3） 通过 executeSql 方法执行查询。

```
ts.executeSql(sqlQuery,[value1,value2, … ],dataHandler,errorHandler)
```

参数说明：

① sqlQuery：需要具体执行的 SQL 语句，可以是 create、select、update、delete。

② [value1，value2，…]：SQL 语句中所有使用到的参数的数组，在 executeSql 方法中，将 SQL 语句中所要使用的参数先用“?”代替，然后依次将这些参数组成数组放在第二个参数中。

③ dataHandler：执行成功时调用的回调函数，通过该函数可以获得查询结果集。

④ errorHandler：执行失败时调用的回调函数。

下面是一个综合的例子，代码如下。

```
<!DOCTYPE HTML>
<html>
<head>
    <title>Web SQL Database</title>
</head>
<body>
  <script type="text/javascript">
    var db = openDatabase('testDB', '1.0', 'Test DB', 2 * 1024 * 1024);
```

```
        var msg;
        db.transaction(function (context) {
         context.executeSql('CREATE TABLE IF NOT EXISTS testTable (id unique, name)');
         context.executeSql('INSERT INTO testTable (id, name) VALUES (0, "Byron")');
         context.executeSql('INSERT INTO testTable (id, name) VALUES (1, "Casper")');
         context.executeSql('INSERT INTO testTable (id, name) VALUES (2, "Frank")');
        });

        db.transaction(function (context) {
        context.executeSql('SELECT * FROM testTable', [], function (context, results)
        {
                var len = results.rows.length, i;
                console.log('Got ' + len + ' rows.');
                   for (i = 0; i < len; i++){
                  console.log('id: ' + results.rows.item(i).id);
                  console.log('name: ' + results.rows.item(i).name);
                }
         });
     });
    </script>
  </body>
  </html>
```

同样用 Chrome 的调试工具可以查看当前浏览器中的 Web SQL 数据，如图 3.3 所示。

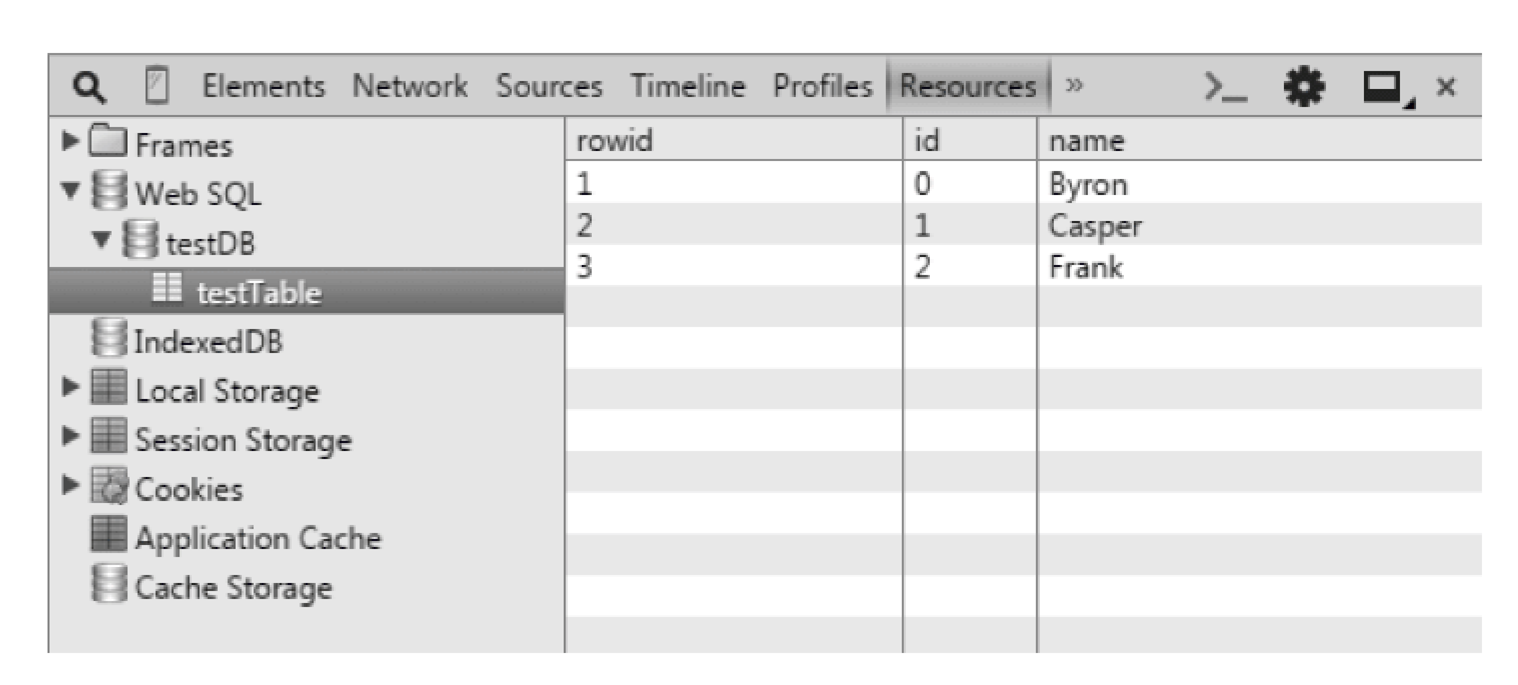


图 3.3　浏览器中的 Web SQL 数据

小　　结

本章主要介绍了 localStorage 数据存储，sessionStorage 数据存储，Web SQL Database 数据存储。通过本章的学习，可以掌握 HTML5 的数据存储方式。

第 4 章 HTML5 离线应用

本章学习目标

- HTML5 离线功能
- 在线状态检测

Web 2.0 技术鼓励个人的参与,每个人都是 Web 内容的撰写者。如果 Web 应用能够提供离线的功能,让用户在没有网络的地方也能进行内容撰写,等到有网络的时候,再同步到 Web 上,就大大方便了用户的使用。HTML5 作为新一代的 HTML 标准,包含对离线功能的支持。本章介绍了 HTML5 离线功能中的离线资源缓存、在线状态检测、本地数据存储等内容,并举例说明了如何使用 HTML5 的新特性开发离线应用。

4.1 HTML5 离线功能介绍

HTML5 是目前正在讨论的新一代 HTML 标准,它代表了现在 Web 领域的最新发展方向。在 HTML5 标准中,加入了新的多样的内容描述标签,直接支持表单验证、视频音频标签、网页元素的拖曳、离线存储和工作线程等功能。其中一个新特性就是对离线应用开发的支持。

在开发支持离线的 Web 应用程序时,开发者通常需要使用以下三个方面的功能。

(1) 离线资源缓存。需要一种方式来指明应用程序离线工作时所需的资源文件。这样,浏览器才能在在线状态时,把这些文件缓存到本地。此后,当用户离线访问应用程序时,这些资源文件会自动加载,从而让用户正常使用。HTML5 中,通过 cache manifest 文件指明需要缓存的资源,并支持自动和手动两种缓存更新方式。

(2) 在线状态检测。开发者需要知道浏览器是否在线,这样才能够针对在线或离线的状态,做出对应的处理。在 HTML5 中,提供了两种检测当前网络是否在线的方式。

(3) 本地数据存储。离线时,需要能够把数据存储到本地,以便在线时同步到服务器上。为了满足不同的存储需求,HTML5 提供了 DOM Storage 和 Web SQL Database 两种存储机制。前者提供了简单的 key/value 的存储方式,而后者提供了基本的关系数据库存储功能。

尽管 HTML5 还处于草稿状态,但是各大主流浏览器都已经实现了其中的很多功能。Chrome、Firefox、Safari 和 Opera 的最新版本都对 HTML5 离线功能提供了完整的支持。IE 8 也支持了其中的在线状态检测和 DOM Storage 功能。下面将具体介绍 HTML5 离线功能中的离线资源缓存、在线状态检测、DOM Storage 和 Web SQL Database,最后通过一

个简单的 Web 程序说明使用 HTML5 开发离线应用的方法。

4.2 离线资源缓存

为了能够让用户在离线状态下继续访问 Web 应用，开发者需要提供一个 cache manifest 文件。这个文件中列出了所有需要在离线状态下使用的资源，浏览器会把这些资源缓存到本地。本节先通过一个例子展示 cache manifest 文件的用途，然后详细描述其书写方法，最后说明缓存的更新方式。

1. cache manifest 示例

下面通过 W3C 提供的示例来说明。Clock Web 应用由三个文件 clock.html、clock.css 和 clock.js 组成。以下是 Clock 应用代码。

```
<!-- clock.html -->
<!DOCTYPE HTML>
<html>
<head>
 <title>Clock</title>
 <script src="clock.js"></script>
 <link rel="stylesheet" href="clock.css">
</head>
<body>
 <p>The time is: <output id="clock"></output></p>
</body>
</html>

/* clock.css */
output { font: 2em sans-serif; }

/* clock.js */
setTimeout(function () {
    document.getElementById('clock').value = new Date();
}, 1000);
```

当用户在离线状态下访问 clock.html 时，页面将无法展现。为了支持离线访问，开发者必须添加 cache manifest 文件，指明需要缓存的资源。这个例子中的 cache manifest 文件为 clock.manifest，它声明了三个需要缓存的资源文件 clock.html、clock.css 和 clock.js。clock.manifest 代码如下。

```
CACHE MANIFEST
clock.html
clock.css
clock.js
```

添加了 cache manifest 文件后，还需要修改 clock.html，把<html>标签的 manifest 属性设置为 clock.manifest。修改后的 clock.html 代码如下。

```
<! -- clock.html -->
<!DOCTYPE HTML>
<html manifest="clock.manifest">
<head>
 <title>Clock</title>
 <script src="clock.js"></script>
 <link rel="stylesheet" href="clock.css">
</head>
<body>
 <p>The time is: <output id="clock"></output></p>
</body>
</html>
```

修改后,当用户在线访问 clock.html 时,浏览器会缓存 clock.html、clock.css 和 clock.js 文件;而当用户离线访问时,这个 Web 应用也可以正常使用了。

2. cache manifest 格式说明

下面说明书写 cache manifest 文件需要遵循的格式。

(1) 首行必须是"CACHE MANIFEST"。

(2) 其后每一行列出一个需要缓存的资源文件名。

(3) 可根据需要列出在线访问的白名单。白名单中的所有资源不会被缓存,在使用时将直接在线访问。声明白名单使用"NETWORK:标识符"。

(4) 如果在白名单后还要补充需要缓存的资源,可以使用"CACHE:标识符"。

(5) 如果要声明某 URI 不能访问时的替补 URI,可以使用"FALLBACK:标识符"。其后的每一行包含两个 URI,当第一个 URI 不可访问时,浏览器将尝试使用第二个 URI。

(6) 注释要另起一行,以 # 号开头。

以下的代码中给出了 cache manifest 中各类标识符的使用示例。

```
CACHE MANIFEST
 # 上一行是必须书写的。

 images/sound-icon.png
 images/background.png

 NETWORK:
 comm.cgi
# 下面是另一些需要缓存的资源,在这个示例中只有一个 css 文件。
 CACHE:
 style/default.css

 FALLBACK:
 /files/projects /projects
```

更新缓存:应用程序可以等待浏览器自动更新缓存,也可以使用 JavaScript 接口手动触发更新。

1. 自动更新

浏览器除了在第一次访问 Web 应用时缓存资源外，只会在 cache manifest 文件本身发生变化时更新缓存。而 cache manifest 中的资源文件发生变化并不会触发更新。

2. 手动更新

开发者也可以使用 window. applicationCache 的接口更新缓存。方法是检测 window . applicationCache. status 的值，若是 UPDATEREADY，那么可以调用 window. application Cache. update()更新缓存。手动更新缓存示范代码如下。

```
if(window.applicationCache.status == window.applicationCache.UPDATEREADY)
 {
    window.applicationCache.update();
 }
```

4.3　在线状态检测

如果 Web 应用程序仅仅是一些静态页面的组合，那么通过 cache manifest 缓存资源文件以后，就可以支持离线访问了。但是随着互联网的发展，特别是 Web 2.0 的概念流行以来，用户提交的数据渐渐成为互联网的主流。那么在开发支持离线的 Web 应用时，就不能仅满足于静态页面的展现，还必须考虑如何让用户在离线状态下也可以操作数据。离线状态时，把数据存储在本地；在线以后，再把数据同步到服务器上。为了做到这一点，开发者首先必须知道浏览器是否在线。HTML5 提供了两种检测是否在线的方式：navigator. online 和 online/offline 事件。

1. navigator. online 事件

navigator. online 属性表示当前是否在线。如果为 true，表示在线；如果为 false，表示离线。当网络状态发生变化时，navigator. online 的值也随之变化。开发者可以通过读取它的值获取网络状态。

2. online/offline 事件

当开发离线应用时，通过 navigator. online 获取网络状态通常是不够的。开发者还需要在网络状态发生变化时立刻得到通知，因此 HTML5 还提供了 online/offline 事件。当在线/离线状态切换时，online/offline 事件将触发在 body 元素上，并且沿着 document. body、document 和 window 的顺序冒泡。因此，开发者可以通过监听它们的 online/offline 事件来获悉网络状态。

4.4　离线应用示例

最后通过一个例子来说明使用 HTML5 开发离线应用的基本方法，这个例子是一个简单的库存管理应用程序，可以用于追踪所拥有的物资。如图 4.1 和图 4.2 所示的是在 Opera Mobile 中测试的结果，屏幕上半段是所有输入项目(书籍、计算机等)的概览。

当用户在清单中选择某一项目时，其明细(Id、Quantity、Name)在表单中部显示。使用 update 按钮可改变所选项目的明细，使用 delete 按钮可从应用程序中删除所选项目。在表

单中输入项目的数量和名称并单击 create 按钮可创建新项目。HTML 页面包含声明、原标记,这些用于移动优化显示、对外部文件(manifest、JavaScript、CSS)的引用,以及组成应用程序基本结构必需的 HTML 元素。

页面的 HTML 代码如下。

```
<!DOCTYPE HTML>
<html manifest="MyHomeStuff.manifest">
<head>
  <meta name="viewport" content="width=480; initial-scale=1.0; maximum-scale=1.0;
user-scalable=0;">
 <title>MyHomeStuff</title>
 <script type="text/javascript" src="MyHomeStuff.js"></script>
</head>
<body onload="onInit()">
  <h3>Overview.</h3>
      <ul id="itemData">
      </ul>
  <h3>Details</h3>
  <form name="itemForm">
   <label for="id">Id:</label><input type="text" name="id" id="id" size=2 disabled=
"true"/>
   <label for="amount">Qty:</label><input type="text" name="amount" id="amount" size = 3/>
  <label for="name">Name:</label><input type="text" name="name" id="name" size=10 /><br>
   <br>
  <input type="button" name="create" value="create" onclick="onCreate()" />
  <input type="button" name="update" value="update" onclick="onUpdate()" />
  <input type="button" name="delete" value="delete" onclick="onDelete()" />
 </form>
 <h4>Status</h4>
 <div id="status"></div>
</body>
</html>
```

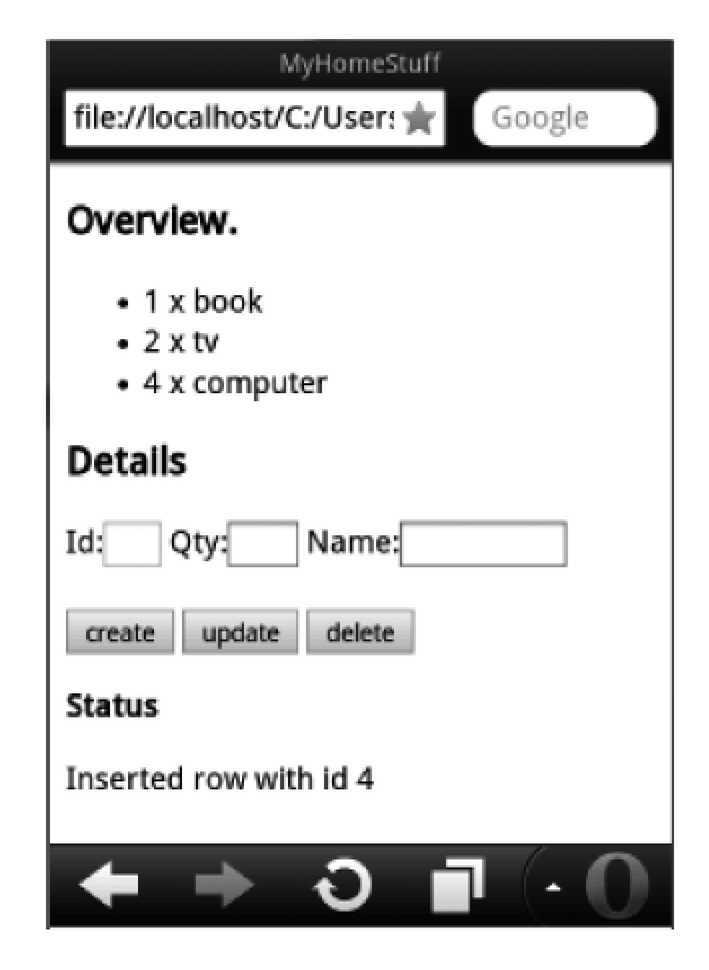


图 4.1　创建新项目

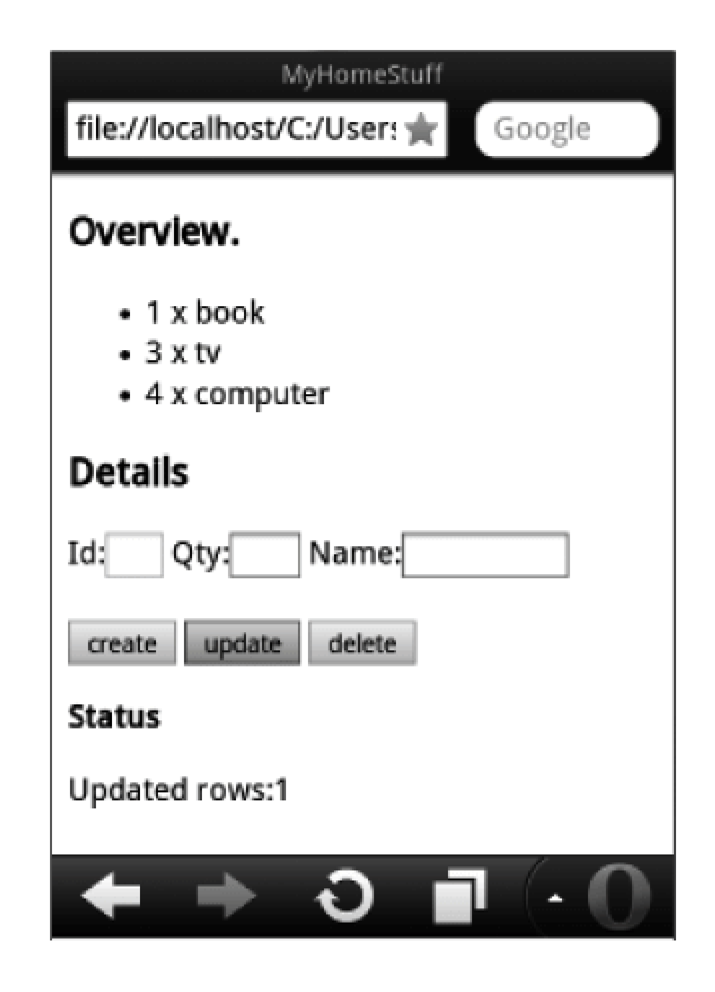


图 4.2　修改所选项目

HTML 元素的事件处理属性指定当页面初始加载(onload)及单击按钮元素(onclick)时执行哪些 JavaScript 函数。

离线 Web 应用程序的 HTML 页面以<!DOCTYPE HTML>标记开始。通过<html manifest="MyHomeStuff.manifest">标记中的清单属性来引用清单。

如上所述,清单指定需要加载到缓存中的文件。本应用程序包含一个 HTML 文件和一个 JavaScript 文件。引用清单的 HTML 文件自动包含在应用程序缓存中。清单只包含以下部分。

manifest 文件如下。

```
CACHE MANIFEST
MyHomeStuff.js
```

JavaScript 代码文件如下。

```
//1.初始化代码：onInit 函数首先检查是否存在强制 openDatabase 函数,如果没有,
//则说明浏览器不支持本地数据库。initDB 函数打开 HTML5 浏览器的数据库。
//成功打开数据库后,执行创建数据库表的 SQL DDL.最后,调用查询现有记录和用数据
//更新 HTML 页面的函数。
var localDB = null;
function onInit(){
    try {
        if (!window.openDatabase) {
            updateStatus("Error: DB not supported");
        }
        else {
            initDB();
            createTables();
            queryAndUpdateOverview();
        }
    }
    catch (e) {
        if (e == 2) {
            updateStatus("Error: Invalid database version.");
        }
        else {
            updateStatus("Error: Unknown error " + e + ".");
        }
        return;
    }
}

function initDB(){
    var shortName = 'stuffDB';
    var version = '1.0';
    var displayName = 'MyStuffDB';
    var maxSize = 65536; //单位为字节
    localDB = window.openDatabase(shortName, version, displayName, maxSize);
```

```
}

function createTables(){
     var query = 'CREATE TABLE IF NOT EXISTS items ( id INTEGER NOT NULL PRIMARY KEY
AUTOINCREMENT, amount VARCHAR NOT NULL, name VARCHAR NOT NULL);';
    try {
        localDB.transaction(function(transaction){
            transaction.executeSql(query, [], nullDataHandler, errorHandler);
            updateStatus("Table 'items' is present");
        });
    }
    catch (e) {
        updateStatus("Error: Unable to create table 'items'" + e + ".");
        return;
    }
}

//2.更新代码：读取和验证表单的字段值。
//如果值有效,则执行更新查询。
//查询结果在更新后的 HTML 页面中显示。
function onUpdate(){
    var id = document.itemForm.id.value;
    var amount = document.itemForm.amount.value;
    var name = document.itemForm.name.value;
    if (amount == "" || name == "") {
        updateStatus("'Amount' and 'Name' are required fields!");
    }
    else {
        var query = "update items set amount = ?, name = ? where id = ?;";
        try {
            localDB.transaction(function(transaction){
                transaction.executeSql(query, [amount, name, id],
                    function(transaction, results){
                    if (!results.rowsAffected) {
                        updateStatus("Error: No rows affected");
                    }
                    else {
                        updateForm("", "", "");
                        updateStatus("Updated rows:" + results.rowsAffected);
                        queryAndUpdateOverview();
                    }
                }, errorHandler);
            });
        }
        catch (e) {
            updateStatus("Error: Unable to perform an UPDATE " + e + ".");
        }
    }
}
```

```
function onDelete(){
    var id = document.itemForm.id.value;
    var query = "delete from items where id = ?;";
    try {
        localDB.transaction(function(transaction){
            transaction.executeSql(query, [id], function(transaction, results){
                if (!results.rowsAffected) {
                    updateStatus("Error: No rows affected.");
                }
                else {
                    updateForm("", "", "");
                    updateStatus("Deleted rows:" + results.rowsAffected);
                    queryAndUpdateOverview();
                }
            }, errorHandler);
        });
    }
    catch (e) {
        updateStatus("Error: Unable to perform an DELETE " + e + ".");
    }

}

function onCreate(){
    var amount = document.itemForm.amount.value;
    var name = document.itemForm.name.value;
    if (amount == "" || name == "") {
        updateStatus("Error: 'Amount' and 'Name' are required fields!");
    }
    else {
        var query = "insert into items (amount, name) VALUES (?, ?);";
        try {
            localDB.transaction(function(transaction){
                transaction.executeSql(query, [amount, name], function(transaction, results){
                    if (!results.rowsAffected) {
                        updateStatus("Error: No rows affected.");
                    }
                    else {
                        updateForm("", "", "");
                        updateStatus("Inserted row with id " + results.insertId);
                        queryAndUpdateOverview();
                    }
                }, errorHandler);
            });
        }
        catch (e) {
            updateStatus("Error: Unable to perform an INSERT " + e + ".");
        }
    }
```

```
}
//选择代码：当用户选取列表元素后执行 onSelect 函数。将用该元素的数据填充明细表单
function onSelect(htmlLIElement){
    var id = htmlLIElement.getAttribute("id");
    query = "SELECT * FROM items where id=?;";
    try {
        localDB.transaction(function(transaction){
            transaction.executeSql(query, [id], function(transaction, results){
                var row = results.rows.item(0);
                updateForm(row['id'], row['amount'], row['name']);

            }, function(transaction, error){
                updateStatus("Error: " + error.code + "<br>Message: " + error.message);
            });
        });
    }
    catch (e) {
        updateStatus("Error: Unable to select data from the db " + e + ".");
    }

}
//概览代码：执行了选择所有数据集的查询。
//对于结果中的每个数据集,创建了 HTML 清单元素并添加到列表中。
//事件处理函数 onSelect,添加到每个列表元素以在点击时响应。
function queryAndUpdateOverview(){
    //remove old table rows
    var dataRows = document.getElementById("itemData").getElementsByClassName("data");
    while (dataRows.length > 0) {
        row = dataRows[0];
        document.getElementById("itemData").removeChild(row);
    };

    //read db data and create new table rows
    var query = "SELECT * FROM items;";
    try {
        localDB.transaction(function(transaction){
            transaction.executeSql(query, [], function(transaction, results){
                for (var i = 0; i < results.rows.length; i++) {

                    var row = results.rows.item(i);
                    var li = document.createElement("li");
                    li.setAttribute("id", row['id']);
                    li.setAttribute("class", "data");
                    li.setAttribute("onclick", "onSelect(this)");

                    var liText = document.createTextNode(row['amount'] + " x " + row['name']);
                    li.appendChild(liText);

                    document.getElementById("itemData").appendChild(li);
```

```
                }
            }, function(transaction, error){
                updateStatus("Error: " + error.code + "<br>Message: " + error.message);
            });
        });
    }
    catch (e) {
        updateStatus("Error: Unable to select data from the db " + e + ".");
    }
}

//3.处理函数代码：它们只作为查询的参数。

errorHandler = function(transaction, error){
    updateStatus("Error: " + error.message);
    return true;
}

nullDataHandler = function(transaction, results){
}

//工具代码：将填充细节表单(updateForm)的字段和状态消息(updateStatus)
function updateForm(id, amount, name){
    document.itemForm.id.value = id;
    document.itemForm.amount.value = amount;
    document.itemForm.name.value = name;
}

function updateStatus(status){
    document.getElementById('status').innerHTML = status;
}
```

小　　结

本章主要介绍了 HTML5 离线功能和在线状态的检测。通过本章的学习，可以开发出 HTML5 的离线应用。

第5章 CSS3 与移动 Web

本章学习目标

- CSS3 的基本特性

CSS2.1 发布至今已经有多年的历史，在这几年里，互联网的发展已经发生了翻天覆地的变化。CSS2.1 有时候难以满足快速提高性能、提升用户体验的 Web 应用的需求。CSS3 标准的出现就是增强 CSS2.1 的功能，减少图片的使用次数以及实现 HTML 页面上的特殊效果。

在 HTML5 逐渐成为 IT 界最热门的话题的同时，CSS3 也开始慢慢地普及起来。目前，很多浏览器都开始支持 CSS3 的部分特性，特别是基于 Webkit 内核的浏览器，其支持力度非常大。在 Android 和 iOS 等移动平台下，正是由于 Apple 和 Google 两家公司大力推广 HTML5 以及各自的 Web 浏览器的迅速发展，CSS3 在移动 Web 浏览器下都能到很好的支持和应用。

CSS3 作为在 HTML 页面中担任页面布局和页面装饰的技术，可以更加有效地对页面布局、字体、颜色、背景或其他动画效果实现精确的控制。

目前，CSS3 是移动 Web 开发的主要技术之一，它在界面修饰方面占有重要的地位。由于移动设备的 Web 浏览器都支持 CSS3，对于不同浏览器之间的兼容性问题，它们之间的差异非常小。不过对于移动 Web 浏览器的某些 CSS 特性，仍然需要做一些兼容性的工作。

当前，CSS3 技术最适合在移动 Web 开发中使用的特性包括：增强的选择器；阴影；强大的背景设置；圆角边框。

接下来的章节将会重点介绍如何使用这些 CSS3 特性来实现移动 Web 界面。

5.1 CSS 盒子模型

CSS3 引入了新的盒模型——弹性盒模型，该模型决定一个盒子在其他盒子中的分布方式以及如何处理可用的空间。这与 XUL(火狐使用的用户交互语言)相似，其他语言也使用相同的盒模型，如 XAML、GladeXML。

使用该模型，可以很轻松地创建自适应浏览器窗口的流动布局或自适应字体大小的弹性布局。本章的例子使用以下的 HTML 代码。

```
<body>
<div id="box1">1</div>
```

```
<div id="box2"> 2 </div>
<div id="box3"> 3 </div>
</body>
```


传统的盒模型基于 HTML 流在垂直方向上排列盒子。使用弹性盒模型可以规定特定的顺序，也可以将其反转。要开启弹性盒模型，只需设置拥有子盒子的盒子的 display 的属性值为 box(或 inline-box)即可。

水平或垂直分布：box-orient 定义分布的坐标轴，vertical 和 horizonal。这两个值定义盒子如何显示，以下是示例代码，显示效果如图 5.1 所示。

```
body{
  display: box;
  box-orient: horizontal;
}
```

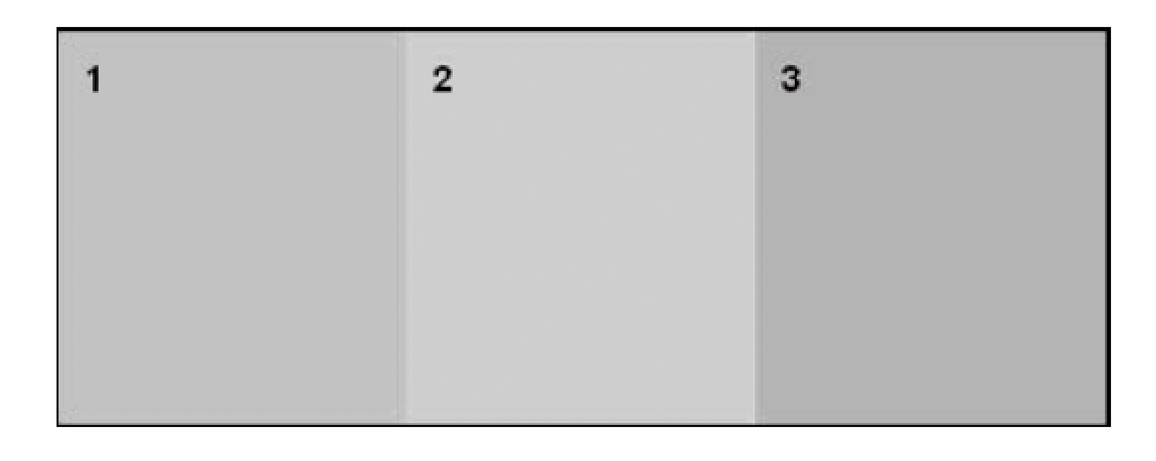


图 5.1　水平分布

反向分布：box-direction 可以设置盒子出现的顺序。默认情况下，只需定义分布坐标轴——box 随 HTML 流分布。如果为水平坐标轴，则从左到右分布；垂直坐标轴则从上到下分布。定义 box-direction 的属性值为 reverse，则反转盒子的排列顺序。以下是示例代码，显示效果如图 5.2 所示。

```
body {
  display: box;
  box-orient: vertical;
  box-direction: reverse;
}
```



图 5.2　反向分布

具体分布：属性 box-ordinal-group 定义盒子分布的顺序。可以随意地控制其分布顺序。这些组以一个从“1”开始的数字定义，盒模型将首先分布这些组，所有这些盒子将在每

个组中。分布将从小到大排列。以下是示例代码，显示效果如图 5.3 所示。

```
body {
display: box;
box-orient: vertical;
box-direction : reverse;
}
#box1 {
box-ordinal-group: 2;
}
#box2 {
box-ordinal-group: 2;
}
#box3 {
box-ordinal-group: 1;
}
```



图 5.3　具体分布

盒子尺寸：默认情况下，盒子并不具有弹性，如果 box-flex 的属性值至少为 1 时，则变得富有弹性。

如果盒子不具有弹性，它将尽可能的拉宽使其内容可见，且没有任何溢出，其大小由 width 和 height 来决定（或 min-height、min-width、max-width、max-height）。

如果盒子是弹性的，其大小将按下面的方式计算：具体的大小声明（width、height、min-width、min-height、max-width、max-height）。

如果盒子没有任何大小声明，那么其大小将完全取决于父 box 的大小。即：盒子的大小等于父级盒子的大小乘以其 box-flex 在所有子盒子 box-flex 总和中的百分比（子盒子的大小=父盒子的大小×子盒子的 box-flex/所有子盒子的 box-flex 值的和）。

如果一个或更多的盒子有一个具体的大小声明，那么其大小将计算其中，余下的弹性盒子将按照上面的原则分享剩下的可利用空间。

下面的例子中，box1 的大小为 box2 的两倍，box2 与 box3 大小一样。看起来好像是用百分比定义盒子的大小，但是有一个区别：使用弹性盒模型，增加一个盒子，无须重新计算其大小。以下是示例代码，显示效果如图 5.4 所示。

```
body {
display: box;
box-orient: horizontal;
}
```

```
#box1 {
box-flex: 2;
}
#box2 {
box-flex: 1;
}

#box3 {
box-flex: 1;
}
```

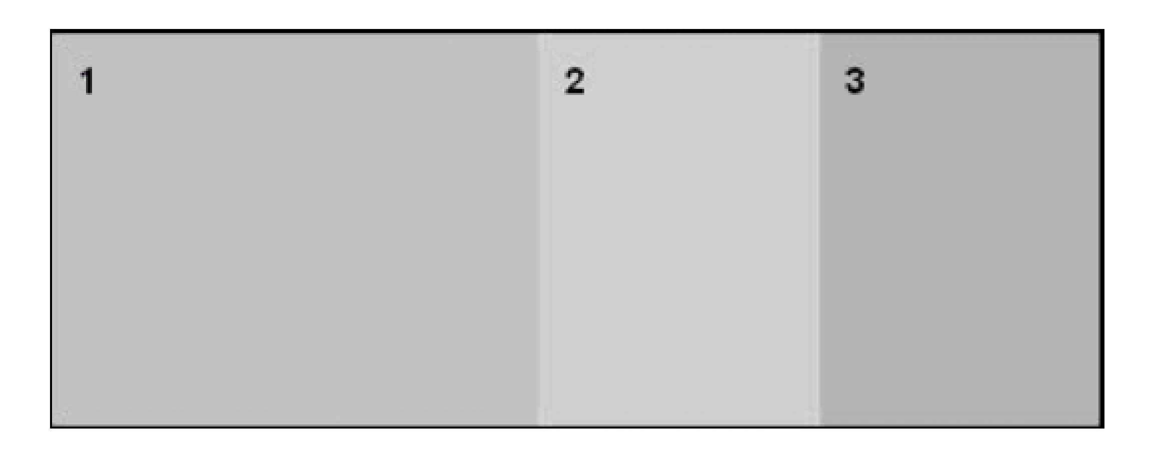


图 5.4　使用弹性盒模型

下面的例子中，box3 并不是弹性的，宽度为 160px；这样 box1 和 box2 将有 240px 的可利用空间。因此，box1 的宽度为 160px(240×2/3)，box2 的宽度为 80px(240×1/3)。以下是示例代码，显示效果如图 5.5 所示。

```
body {
display: box;
box-orient: horizontal;
width: 400px;
}
#box1 {
box-flex: 2;
}
#box2 {
box-flex: 1;
}
#box3 {
width: 160px;
}
```

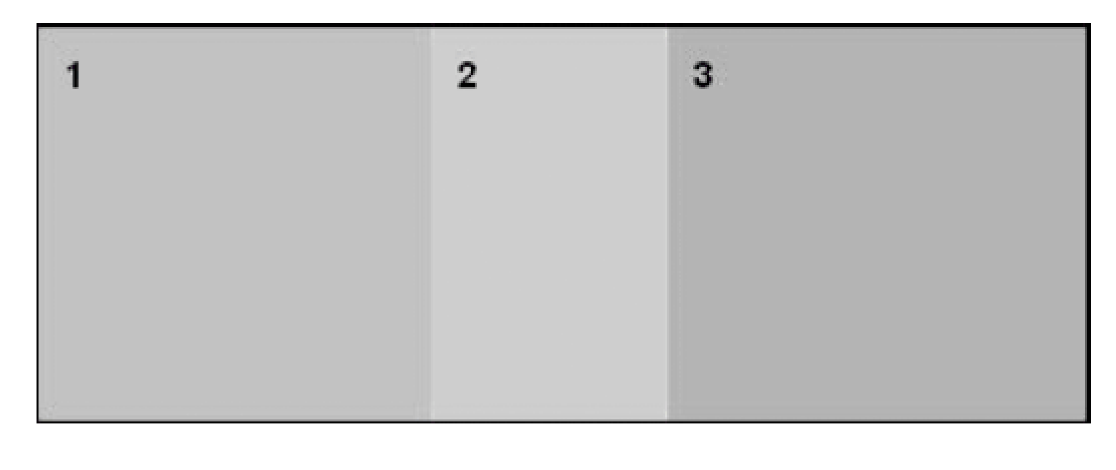


图 5.5　box3 不是弹性的情况

溢出管理：因为是弹性盒子、非弹性盒子混排，有可能所有盒子的尺寸大于或小于父盒子的尺寸。这样就有可能空间太多或空间不足。

空间太多的处理方法：可利用空间的分布取决于两个属性值 box-align 和 box-pack。

属性 box-pack 管理水平方向上的空间分布，有以下 4 个可能属性：start、end、justify 和 center。

(1) start——所有盒子在父盒子的左侧，余下的空间在右侧。

(2) end——所有盒子在父盒子的右侧，余下的空间在左侧。

(3) justify——余下的空间在盒子间平均分配。

(4) center——可利用的空间在父盒子的两侧平均分配。

属性 box-align 管理垂直方向上的空间分布，有以下 5 个可能属性：start、end、center、baseline 和 stretch。

(1) start——每个盒子沿父盒子的上边缘排列，余下的空间位于底部。

(2) end——每个盒子沿父盒子的下边缘排列，余下的空间位于顶部。

(3) center——可用空间平均分配，上面一半，下面一半。

(4) baseline——所有盒子沿着它们的基线排列，余下的空间可前可后。

(5) stretch——每个盒子的高度调整到适合父盒子的高度。

空间不足的处理方法：与传统的盒模型一样，overflow 属性用来决定其显示方式。为了避免溢出，可以设置 box-lines 为 multiple 使其换行显示。

弹性盒模型看起来很不错，Gecko 和 Webkit 对该模型都有一些尝试性的测试。在这些属性之前加上-moz 和-webkit 即可使用该属性。也即是说，Firefox、Safari、Chrome 可以使用这些特性。

作为前端开发者来说，该模型对解决网页设计中一些常见的问题非常方便，如表单布局、垂直居中、视觉上分离 HTML 流等。不久的将来它将成为一个 Web 标准。

5.2 选　择　器

在 CSS2 中经常使用三个简单的选择器：# ID、. class、标签选择器，但是随着 CSS3 的到来，作为前端开发者还需要掌握下面 30 个基本的选择器，这样才可以在平时开发中得心应手。

1. *：通用元素选择器

* 选择器是选择页面上的全部元素，下面代码的作用是把全部元素的 margin 和 padding 设为 0，是最基本的清除默认 CSS 样式的方法。

```
* {
 margin: 0;
 padding: 0;
}

* 选择器也可以应用到子选择器中，例如下面的代码。
#container * {
```

```
border: 1px solid black;
}
```

这样 ID 为 container 的所有子标签元素都被选中了，并且设置了 border。通用元素选择器兼容的浏览器有：IE 6+、Firefox、Chrome、Safari、Opera。

2. #ID：ID 选择器

ID 选择器是 CSS 中效率最高的选择器，使用的时候要保证 ID 的唯一性。ID 选择器兼容的浏览器有：IE 6+、Firefox、Chrome、Safari、Opera。例如下面的代码。

```
#container {
   width: 960px;
   margin: auto;
}
```

3. .class：类选择器

类选择器效率低于 ID 选择器，一个页面可以有多个 class，并且 class 可以放在不同的标签中使用。例如下面的代码。

```
.error {
  color: red;
}
```

类选择器兼容的浏览器有：IE 6+、Firefox、Chrome、Safari、Opera。

4. X Y：标签组合选择器

标签组合选择器是常用的选择器。例如下面的代码。

```
li a {
  text-decoration: none;
}
```

标签组合选择器兼容的浏览器有：IE 6+、Firefox、Chrome、Safari、Opera。

5. X：标签选择器

标签选择器也是常用的选择器，如果只是想要页面中的某个标签样式改变，可以选择使用标签选择器。例如下面的代码。

```
a { color: red; }
ul { margin-left: 0; }
```

标签选择器兼容的浏览器有：IE 6+、Firefox、Chrome、Safari、Opera。

6. 伪类选择器

伪类选择器，最常用的为 a 标签。例如下面的代码。

```
a:link { color: red; }
a:visted { color: purple; }
```

伪类选择器兼容的浏览器有：IE 7+、Firefox、Chrome、Safari、Opera。

7. X + Y：毗邻元素选择器

毗邻元素选择器，匹配的是所有紧随 X 元素之后的同级元素 Y。例如下面的代码。

```
ul + p {
   color: red;
}
```

兼容的浏览器有：IE 7+、Firefox、Chrome、Safari、Opera。

8. X > Y：子元素选择器

下面的代码是子元素选择器的例子。

```
div#container > ul {
  border: 1px solid black;
}
```

在此代码中，匹配#container 下的所有子元素。

关于 X>Y 和 X Y 的区别请再看下面的 HTML 实例。

```
<div id="container">
   <ul>
      <li> List Item
        <ul>
           <li> Child </li>
        </ul>
      </li>
      <li> List Item </li>
      <li> List Item </li>
      <li> List Item </li>
   </ul>
</div>
```

选择器#container > ul 只会匹配到第一个 ul，也就是#container 的子元素 ul，而不是 li 里面的 ul，但是 div ul 则可以匹配到所有 div 里面的 ul。

子元素选择器兼容的浏览器有：IE 7+、Firefox、Chrome、Safari、Opera。

9. X ~ Y：

这种格式的例子如下。

```
ul ~ p {
   color: red;
}
```

在此代码中，匹配任何在 ul 元素之后的同级 p 元素。也就是选择了 ul 之后的同级所有的元素。兼容的浏览器有：IE 7+、Firefox、Chrome、Safari、Opera。

10. X[title]：属性选择器

这种格式的例子如下。

```
a[title] {
    color: green;
}
```


在此代码中，匹配具有某属性的标签，例如，上例中是匹配具有 title 属性的 a 标签。兼容的浏览器有：IE 7+、Firefox、Chrome、Safari、Opera。

11. X[href="foo"]

这种格式如以下的代码。

```
a[href = "http://js8.in"] {
  color: #1f6053; /* nettuts green */
}
```

这种格式其实也属于属性选择器，匹配属性中为某个值的标签。例如，上例中匹配的为 href="http://js8.in"的 a 标签，而其他链接的 a 标签不选择。兼容的浏览器有：IE 7+、Firefox、Chrome、Safari、Opera。

12. X[href *="nettuts"]

这种格式如以下的代码。

```
a[href * = "tuts"] {
  color: #1f6053; /* nettuts green */
}
```

这种格式其实也属于属性选择器，匹配 href 中所有含有 tuts 的标签(正则匹配)。兼容的浏览器有：IE 7+、Firefox、Chrome、Safari、Opera。

13. X[href ^="http"]

这种格式如以下的代码。

```
a[href ^ = "http"] {
    background: url(path/to/external/icon.png) no-repeat;
    padding-left: 10px;
}
```

与上面的属性选择标签类似，但是匹配的是以 http 开头的 a 标签(正则匹配)。兼容的浏览器有：IE 7+、Firefox、Chrome、Safari、Opera。

14. X[href $ =".jpg"]

这种格式如以下的代码。

```
a[href $ = ".jpg"] {
    color: red;
}
```

匹配属性中以.jpg 结尾的标签(正则匹配)，这种格式其实也属于属性选择器，兼容的浏览器有：IE 7+、Firefox、Chrome、Safari、Opera。

15. X[data－ * ＝"foo"]

如果要匹配所有的图片链接，可以通过下面的 CSS 来实现。

```
a[href $ =".jpg"],
a[href $ =".jpeg"],
a[href $ =".png"],
a[href $ =".gif"] {
   color: red;
}
```

但是如果给 a 标签添加一个 data-filetype 属性，就可以使用下面的 CSS 来快速地选择需要匹配的标签了。

```
<a href="path/to/image.jpg" data-filetype="image"> Image Link </a>
<pre lang="css"> a[data-filetype="image"] {
   color: red;
}
```

这种格式兼容的浏览器有：IE 7＋、Firefox、Chrome、Safari、Opera。

16. X[foo～＝"bar"]

这种格式如以下的代码。

```
a[data-info~="external"] {
   color: red;
}

a[data-info~="image"] {
   border: 1px solid black;
}
```

匹配属性中具有多个空格分隔的值、其中一个值等于"bar"的 X 元素。

这种格式兼容的浏览器有：IE 7＋、Firefox、Chrome、Safari、Opera。

17. X:checked

这种格式如以下的代码。

```
input[type=radio]:checked {
   border: 1px solid black;
}
```

这个选择器主要用于 checkbox，选择 checkbox 为当前选中的那个标签。

这种格式兼容的浏览器有：IE 7＋、Firefox、Chrome、Safari、Opera。

18. X:after

这种格式如以下的代码。

```
.clearfix:after {
    content: "";
```

```
        display: block;
        clear: both;
        visibility: hidden;
        font - size: 0;
        height: 0;
        }

.clearfix {
    * display: inline - block;
    _height: 1 % ;
}
```

before 和 after 是在选择的标签之前或者之后插入内容，一般用于清除浮动，但是对于 IE 6、IE 7 是不可用的。这种格式兼容的浏览器有：IE 7＋、Firefox、Chrome、Safari、Opera。

19. X:hover

这种格式如以下的代码。

```
div:hover {
  background: #e3e3e3;
}
```

最常用的就是 a 标签了，但是在 IE 6 浏览器下除了 a 标签之外，其他标签 div:hover 不匹配。这种格式兼容的浏览器有：IE 6＋、Firefox、Chrome、Safari、Opera。

20. X:not(selector)

这种格式如以下的代码。

```
* :not(p) {
  color: green;
}
```

选择除了( )中选择器之外的标签元素。

这种格式兼容的浏览器有：IE 9、Firefox、Chrome、Safari、Opera。

21. X::pseudoElement

这种格式如以下的代码。

```
p::first - line {
    font - weight: bold;
    font - size: 1.2em;
}
p::first - letter {
    float: left;
    font - size: 2em;
    font - weight: bold;
    font - family: cursive;
    padding - right: 2px;
}
```

分别用于匹配元素的第一行和第一个字母。这种格式兼容的浏览器有：IE 6+、Firefox、Chrome、Safari、Opera。

22. X:nth-child(n)

这种格式如以下的代码。

```
li:nth-child(3) {
    color: red;
}
```

匹配 X 元素中从头数第几个标签，例如，上面的代码是匹配第三个 li 标签。这种格式兼容的浏览器有：IE 9、Firefox 3.5+、Chrome、Safari、Opera。

23. X:nth-last-child(n)

这种格式如以下的代码。

```
li:nth-last-child(2) {
    color: red;
}
```

与上一个选择器相反，这个选择器是倒序匹配第几个标签。上面的代码的意思是匹配倒数第二个 li 标签。这种格式兼容的浏览器有：IE 9、Firefox 3.5+、Chrome、Safari、Opera。

24. X:nth-of-type(n)

这种格式如以下的代码。

```
ul:nth-of-type(3) {
    border: 1px solid black;
}
```

与:nth-child()作用类似，但是仅匹配使用同种标签的元素。这种格式兼容的浏览器有：IE 9、Firefox 3.5+、Chrome、Safari、Opera。

25. X:nth-last-of-type(n)

这种格式如以下的代码。

```
ul:nth-last-of-type(3) {
    border: 1px solid black;
}
```

与:nth-last-child()作用类似，但是仅匹配使用同种标签的元素。这种格式兼容的浏览器有：IE 9、Firefox 3.5+、Chrome、Safari、Opera。

26. X:first-child

这种格式如以下的代码。

```
ul li:first-child {
    border-top: none;
}
```

匹配其父元素的第 n 个子元素，第一个编号为 1。这种格式兼容的浏览器有：IE 7+、Firefox、Chrome、Safari、Opera。

27. X：last-child

这种格式如以下的代码。

```
ul > li:last - child {
    color: green;
}
```

匹配其父元素的倒数第 n 个子元素，第一个编号为 1。这种格式兼容的浏览器有：IE 9、Firefox、Chrome、Safari、Opera。

28. X：only-child

这种格式如以下的代码。

```
div p:only - child {
    color: red;
}
```

匹配父元素下仅有的一个子元素，等同于：first-child：last-child 或：nth-child（1）：nth-last-child（1）。这种格式兼容的浏览器有：IE 9、Firefox、Chrome、Safari、Opera。

29. X：only-of-type

这种格式如以下的代码。

```
li:only - of - type {
    font - weight: bold;
}
```

匹配父元素下使用同种标签的唯一一个子元素，等同于：first-of-type：last-of-type 或：nth-of-type（1）：nth-last-of-type（1）。这种格式兼容的浏览器有：IE 9、Firefox3.5+、Chrome、Safari、Opera。

30. X：first-of-type

这种格式如以下的代码。

```
li:only - of - type {
    font - weight: bold;
}
```

匹配父元素下使用同种标签的第一个子元素，等同于：nth-of-type（1）。这种格式兼容的浏览器有：IE 9、Firefox 3.5+、Chrome、Safari、Opera。

5.3 边　　框

通过 CSS3 可以创建圆角边框，向矩形添加阴影，使用图片来绘制边框，并且不需使用设计软件，比如 Photoshop。在本节中将学到以下边框属性：border-radius、box-shadow 和

border-image。

IE 9 支持 border-radius 和 box-shadow。Firefox 支持所有新的边框属性。Chrome 和 Safari 支持 border-radius 和 box-shadow，但是对 border-image 的支持需要加前缀-webkit-。Opera 支持 border-radius 和 box-shadow，但是对 border-image 的支持需要加前缀-o-。

1. CSS3 圆角边框

在 CSS2 中添加圆角矩形需要技巧，必须为每个圆角使用不同的图片。在 CSS3 中，创建圆角是非常容易的，CSS3 中的 border-radius 属性用于创建圆角。

实例：向 div 元素中添加圆角。

```
div
{
border:2px solid;
border-radius:25px;
-moz-border-radius:25px; /* Old Firefox */
}
```

2. CSS3 边框阴影

在 CSS3 中，box-shadow 用于向方框中添加阴影。

实例：向 div 元素中添加方框阴影。

```
div
{
box-shadow: 10px 10px 5px #888888;
}
```

3. CSS3 边框图片

通过 CSS3 的 border-image 属性，可以使用图片来创建边框：border-image 属性可以规定用于边框的图片。用于创建上面的边框的原始图片如图 5.6 所示。



图 5.6　边框图片

实例：使用图片创建围绕 div 元素的边框。

```
div
{
border-image:url(border.png) 30 30 round;
-moz-border-image:url(border.png) 30 30 round;       /* 老的 Firefox */
-webkit-border-image:url(border.png) 30 30 round;    /* Safari 和 Chrome */
-o-border-image:url(border.png) 30 30 round;         /* Opera */
}
```

CSS3 新的边框属性如表 5.1。

表 5.1　CSS3 新的边框属性

属　　性	描　　述
border-image	设置所有 border-image-* 属性的简写属性
border-radius	设置所有 4 个 border-*-radius 属性的简写属性
box-shadow	向方框添加一个或多个阴影

5.4　CSS3 背景

CSS3 包含多个新的背景属性，它们提供了对背景更强大的控制。在本节将学到以下背景属性：background-size、background-origin。

Firefox 3.6 以及更早的版本不支持 background-origin 属性，对 background-size 的支持需要加前缀-moz-。Safari 4 对新背景属性的支持需要加前缀-webkit-。IE 9、Firefox 4、Chrome、Safari 5 以及 Opera 支持新的背景属性。

1. CSS3 background-size 属性

background-size 属性规定背景图片的尺寸。在 CSS3 之前，背景图片的尺寸是由图片的实际尺寸决定的。在 CSS3 中，可以规定背景图片的尺寸，这就允许在不同的环境中重复使用背景图片。能够以像素或百分比规定尺寸。如果以百分比规定尺寸，那么尺寸是相对于父元素的宽度和高度。

例如，调整背景图片的大小，代码如下。

```
div
{
background:url(bg_flower.gif);
-moz-background-size:63px 100px;        /* 老版本的 Firefox */
background-size:63px 100px;
background-repeat:no-repeat;
}
```

例如，对背景图片进行拉伸，使其完成填充内容区域，代码如下。

```
div
{
background:url(bg_flower.gif);
-moz-background-size:40% 100%;        /* 老版本的 Firefox */
background-size:40% 100%;
background-repeat:no-repeat;
}
```

2. CSS3 background-origin 属性

background-origin 属性规定背景图片的定位区域。背景图片可以放置于 content-box、padding-box 或 border-box 区域，如图 5.7 所示。

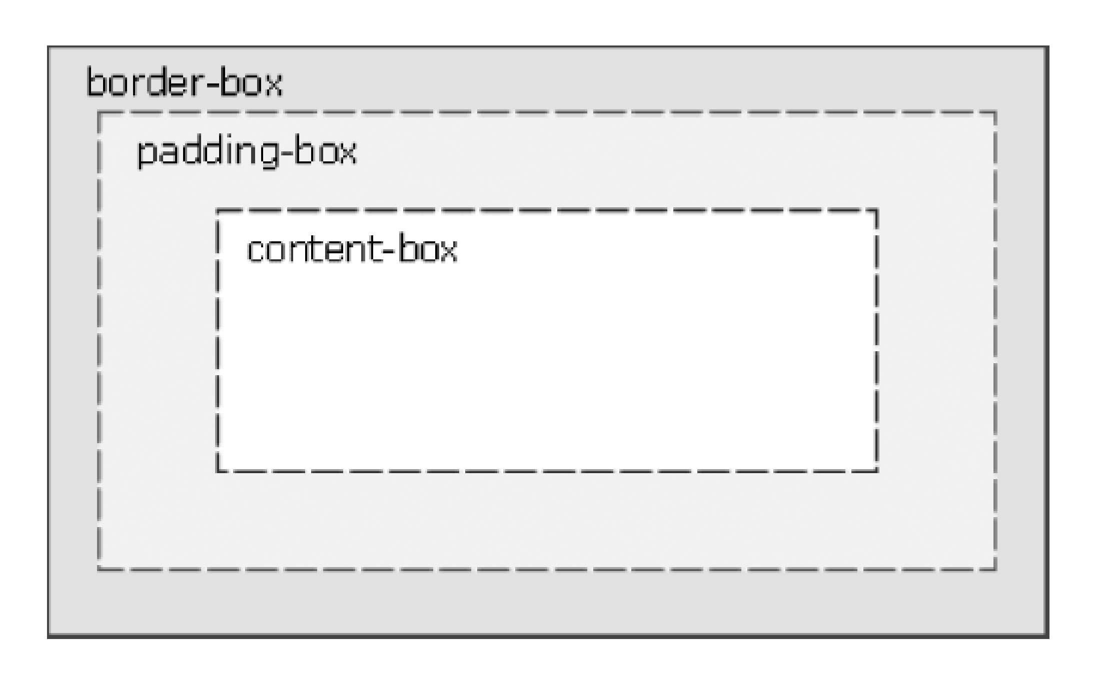


图 5.7 background-origin 属性规定背景图片的区域

例如,在 content-box 中定位背景图片,代码如下。

```
div
{
background:url(bg_flower.gif);
background-repeat:no-repeat;
background-size:100% 100%;
-webkit-background-origin:content-box; /* Safari */
background-origin:content-box;
}
```

3. CSS3 多重背景图片

CSS3 可以为元素使用多个背景图像。

例如,为 body 元素设置两幅背景图片,代码如下。

```
body
{
background-image:url(bg_flower.gif),url(bg_flower_2.gif);
}
```

CSS3 新的背景属性如表 5.2 所示。

表 5.2 CSS3 新的背景属性

属　　性	描　　述
background-clip	规定背景的绘制区域
background-origin	规定背景图片的定位区域
background-size	规定背景图片的尺寸

5.5 CSS3 文本效果

CSS3 包含多个新的文本特性。在本节中将学到如下文本属性：text-shadow、word-wrap 和 text-decoration。

IE 仍然不支持 text-shadow 属性。Firefox、Chrome、Safari 以及 Opera 支持 text-

shadow 属性。浏览器主流浏览器都支持 word-wrap 属性。

1. CSS3 文本阴影

在 CSS3 中，text-shadow 可向文本应用阴影，如图 5.8 所示。

文本阴影效果！

图 5.8　CSS3 文本应用阴影

还可以规定水平阴影、垂直阴影、模糊距离，以及阴影的颜色。下面的代码是向标题添加阴影。

```
h1
{
text - shadow: 5px 5px 5px #FF0000;
}
```

2. CSS3 自动换行

如果单词太长的话就可能超出某个区域。例如：

```
This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long
word will break and wrap to the next line.
```

在 CSS3 中，word-wrap 属性可以强制文本进行换行，这意味着会对单词进行拆分。如下面的 CSS 代码允许对长单词进行拆分，并换行到下一行。

```
p {word - wrap:break - word;}
```

3. 文字渲染

CSS3 里面开始支持对文字的更深层次的渲染，如下面的例子。

```
div {
- webkit - text - fill - color: black;
- webkit - text - stroke - color: red;
- webkit - text - stroke - width: 2.75px;
}
```

这里主要以 Webkit 内核浏览器为例，效果如图 5.9 所示。

文本渲染效果

图 5.9　CSS3 文字渲染

text-fill-color：文字内部填充颜色。

text-stroke-color：文字边界填充颜色。

text-stroke-width：文字边界宽度。

CSS3 新的文本属性如表 5.3 所示。

表 5.3　CSS3 新的文本属性

属　　性	描　　述
hanging-punctuation	规定标点字符是否位于线框之外
punctuation-trim	规定是否对标点字符进行修剪
text-align-last	设置如何对齐最后一行或紧挨着强制换行符之前的行
text-emphasis	向元素的文本应用重点标记以及重点标记的前景色
text-justify	规定当 text-align 设置为 justify 时所使用的对齐方法
text-outline	规定文本的轮廓
text-overflow	规定当文本溢出包含元素时发生的操作
text-shadow	向文本添加阴影
text-wrap	规定文本的换行规则
word-break	规定非中日韩文本的换行规则
word-wrap	允许对长的不可分割的单词进行分割并换行到下一行

5.6　CSS3 字体

在 CSS3 之前，Web 设计师必须使用已在用户计算机上安装好的字体。通过 CSS3，Web 设计师可以使用他们喜欢的任意字体。当找到或购买到希望使用的字体时，可将该字体文件存放到 Web 服务器上，它会在需要时被自动下载到用户的计算机上。

用户自己的字体是在 CSS3 @font-face 规则中定义的。Firefox、Chrome、Safari 以及 Opera 支持.ttf(True Type Fonts)和.otf(OpenType Fonts)类型的字体。

IE 9+支持新的@font-face 规则，但只支持.eot 类型的字体(Embedded OpenType)。

注释：IE 8 以及更早的版本不支持新的@font-face 规则。

在新的@font-face 规则中，必须首先定义字体的名称(比如 myFirstFont)，然后指向该字体文件。如需为 HTML 元素使用字体，请通过 font-family 属性来引用字体的名称(myFirstFont)。

例：

```
<style>
@font-face
{
font-family: myFirstFont;
src: url('Sansation_Light.ttf'),
     url('Sansation_Light.eot');        /* IE 9+ */
}

div
{
font-family:myFirstFont;
}
</style>
```

例：使用粗体字体时必须为粗体文本添加另一个包含描述符的@font-face。

```
@font-face
{
font-family: myFirstFont;
src: url('Sansation_Bold.ttf'),
     url('Sansation_Bold.eot');        /* IE 9+ */
font-weight:bold;
}
```

文件 Sansation_Bold.ttf 是另一个字体文件，它包含 Sansation 字体的粗体字符。只要 font-family 为 myFirstFont 的文本需要显示为粗体，浏览器就会使用该字体。通过这种方式，可以为相同的字体设置许多@font-face 规则。

表 5.4 中列出了能够在@font-face 规则中定义的所有字体描述符。

表 5.4　CSS3 字体描述符

描述符	值	描述
font-family	name	必需。规定字体的名称
src	URL	必需。定义字体文件的 URL
font-stretch	• normal • condensed • ultra-condensed • extra-condensed • semi-condensed • expanded • semi-expanded • extra-expanded • ultra-expanded	可选。定义如何拉伸字体。默认是 normal
font-style	• normal • italic • oblique	可选。定义字体的样式。默认是 normal
font-weight	• normal • bold • 100 • 200 • 300 • 400 • 500 • 600 • 700 • 800 • 900	可选。定义字体的粗细。默认是 normal
unicode-range	unicode-range	可选。定义字体支持的 Unicode 字符范围。默认是“U+0－10FFFF”

5.7 CSS3 2D 转换

通过 CSS3 转换，能够对元素进行移动、缩放、转动、拉长或拉伸。转换是使元素改变形状、尺寸和位置的一种效果。可以使用 2D 或 3D 转换来转换元素。

IE 9 需要加前缀-ms-。Firefox 需要加前缀-moz-。Chrome 和 Safari 需要加前缀-webkit-。Opera 需要加前缀-o-。

在本节中，将学到如下 2D 转换方法。

(1) translate()：移动。

(2) rotate()：旋转。

(3) scale()：绽放。

(4) skew()：以给定的角度转动。

(5) matrix()：旋转、缩放、移动以及倾斜元素。

例：

```
div
{
transform: rotate(30deg);
-ms-transform: rotate(30deg);          /* IE 9 */
-webkit-transform: rotate(30deg);      /* Safari and Chrome */
-o-transform: rotate(30deg);           /* Opera */
-moz-transform: rotate(30deg);         /* Firefox */
}
```

(1) translate()方法：通过 translate()方法，根据给定的 left(x 坐标)和 top(y 坐标)位置参数，元素从其当前位置移动。

例：

```
div
{
transform: translate(50px,100px);
-ms-transform: translate(50px,100px);          /* IE 9 */
-webkit-transform: translate(50px,100px);      /* Safari and Chrome */
-o-transform: translate(50px,100px);           /* Opera */
-moz-transform: translate(50px,100px);         /* Firefox */
}
```

值 translate(50px,100px)把元素从左侧移动 50 像素，从顶端移动 100 像素。

(2) rotate()方法：通过 rotate()方法，元素顺时针旋转给定的角度。允许负值，元素将逆时针旋转。

例：

```
div
{
```

```
transform: rotate(30deg);
-ms-transform: rotate(30deg);           /* IE 9 */
-webkit-transform: rotate(30deg);       /* Safari and Chrome */
-o-transform: rotate(30deg);            /* Opera */
-moz-transform: rotate(30deg);          /* Firefox */
}
```

值 rotate(30deg)表示把元素顺时针旋转 30°。

(3) scale()方法：通过 scale()方法，根据给定的宽度(X 轴)和高度(Y 轴)参数，元素的尺寸会增加或减少。

例：

```
div
{
transform: scale(2,4);
-ms-transform: scale(2,4);          /* IE 9 */
-webkit-transform: scale(2,4);      /* Safari 和 Chrome */
-o-transform: scale(2,4);           /* Opera */
-moz-transform: scale(2,4);         /* Firefox */
}
```

值 scale(2,4)表示把宽度转换为原始尺寸的 2 倍，把高度转换为原始高度的 4 倍。

(4) skew()方法：通过 skew()方法，根据给定的水平线(X 轴)和垂直线(Y 轴)参数，元素转动给定的角度。

例：

```
div
{
transform: skew(30deg,20deg);
-ms-transform: skew(30deg,20deg);          /* IE 9 */
-webkit-transform: skew(30deg,20deg);      /* Safari and Chrome */
-o-transform: skew(30deg,20deg);           /* Opera */
-moz-transform: skew(30deg,20deg);         /* Firefox */
}
```

值 skew(30deg,20deg)表示围绕 X 轴把元素转动 30°，围绕 Y 轴转动 20°。

(5) matrix()方法：matrix()方法把所有 2D 转换方法组合在一起，需要 6 个参数，包括数学函数，可以旋转、缩放、移动以及倾斜元素。

例如，使用 matrix 方法将 div 元素旋转 30°。

```
div
{
transform:matrix(0.866,0.5,-0.5,0.866,0,0);
-ms-transform:matrix(0.866,0.5,-0.5,0.866,0,0);         /* IE 9 */
-moz-transform:matrix(0.866,0.5,-0.5,0.866,0,0);        /* Firefox */
-webkit-transform:matrix(0.866,0.5,-0.5,0.866,0,0);     /* Safari and Chrome */
-o-transform:matrix(0.866,0.5,-0.5,0.866,0,0);          /* Opera */
}
```

表 5.5 中列出了所有的转换属性。

表 5.5 转换属性

属 性	描 述
transform	向元素应用 2D 或 3D 转换
transform-origin	允许改变被转换元素的位置

2D Transform 方法如表 5.6 所示。

表 5.6 转换方法

属 性	描 述
matrix(n,n,n,n,n,n)	定义 2D 转换,使用 6 个值的矩阵
translate(x,y)	定义 2D 转换,沿着 X 和 Y 轴移动元素
translateX(n)	定义 2D 转换,沿着 X 轴移动元素
translateY(n)	定义 2D 转换,沿着 Y 轴移动元素
scale(x,y)	定义 2D 缩放转换,改变元素的宽度和高度
scaleX(n)	定义 2D 缩放转换,改变元素的宽度
scaleY(n)	定义 2D 缩放转换,改变元素的高度
rotate(angle)	定义 2D 旋转,在参数中规定角度
skew(x-angle,y-angle)	定义 2D 倾斜转换,沿着 X 和 Y 轴
skewX(angle)	定义 2D 倾斜转换,沿着 X 轴
skewY(angle)	定义 2D 倾斜转换,沿着 Y 轴

5.8 CSS3 3D 转换

CSS3 可以使用 3D 转换来对元素进行格式化。在本节中,将学到其中的一些 3D 转换方法:rotateX(),rotateY()。

转换是使元素改变形状、尺寸和位置的一种效果。可以使用 2D 或 3D 转换来转换元素。

IE 和 Opera 仍然不支持 3D 转换(仅支持 2D 转换)。Firefox 需要加前缀-moz-。Chrome 和 Safari 需要加前缀-webkit-。

(1) rotateX()方法:通过 rotateX()方法,元素围绕其 X 轴以给定的度数进行旋转。

例:

```
div
{
transform: rotateX(120deg);
-webkit-transform: rotateX(120deg);          /* Safari 和 Chrome */
-moz-transform: rotateX(120deg);             /* Firefox */
}
```

(2) rotateY()旋转：通过 rotateY()方法，元素围绕其 Y 轴以给定的度数进行旋转。

例：

```
div
{
transform: rotateY(130deg);
-webkit-transform: rotateY(130deg);     /* Safari 和 Chrome */
-moz-transform: rotateY(130deg);        /* Firefox */
}
```

表 5.7 中列出了所有的转换属性。

表 5.7　转换属性

属　　性	描　　述
transform	向元素应用 2D 或 3D 转换
transform-origin	允许改变被转换元素的位置
transform-style	规定被嵌套元素如何在 3D 空间中显示
perspective	规定 3D 元素的透视效果
perspective-origin	规定 3D 元素的底部位置
backface-visibility	定义元素在不面对屏幕时是否可见

转换方法如表 5.8 所示。

表 5.8　转换方法

函　　数	描　　述
matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)	定义 3D 转换，使用 16 个值的 4×4 矩阵
translate3d(x,y,z)	定义 3D 转化
translateX(x)	定义 3D 转化，仅使用用于 X 轴的值
translateY(y)	定义 3D 转化，仅使用用于 Y 轴的值
translateZ(z)	定义 3D 转化，仅使用用于 Z 轴的值
scale3d(x,y,z)	定义 3D 缩放转换
scaleX(x)	定义 3D 缩放转换，通过给定一个 X 轴的值
scaleY(y)	定义 3D 缩放转换，通过给定一个 Y 轴的值
scaleZ(z)	定义 3D 缩放转换，通过给定一个 Z 轴的值
rotate3d(x,y,z,angle)	定义 3D 旋转
rotateX(angle)	定义沿 X 轴的 3D 旋转
rotateY(angle)	定义沿 Y 轴的 3D 旋转
rotateZ(angle)	定义沿 Z 轴的 3D 旋转
perspective(n)	定义 3D 转换元素的透视视图

5.9　CSS3 过渡

通过 CSS3 可以在不使用 Flash 动画或 JavaScript 的情况下，当元素从一种样式变换为另一种样式时为元素添加效果。

IE 不支持 transition 属性。Firefox 4 需要加前缀-moz-。Chrome 和 Safari 需要加前缀-webkit-。Opera 需要加前缀-o-。

CSS3 过渡是元素从一种样式逐渐改变为另一种的效果。要实现这一点，必须规定以下两项内容。

(1) 规定把效果添加到哪个 CSS 属性上。

(2) 规定效果的时长。

例：应用于宽度属性的过渡效果，时长为 2s。

```
div
{
transition: width 2s;
-moz-transition: width 2s;          /* Firefox 4 */
-webkit-transition: width 2s;       /* Safari 和 Chrome */
-o-transition: width 2s;            /* Opera */
}
```

注释：如果时长未规定，则不会有过渡效果，因为默认值是 0。

例：规定当鼠标指针悬浮于<div>元素上时。

```
div:hover
{
width:300px;
}
```

注释：当指针移出元素时，它会逐渐变回原来的样式。

多项改变：如需向多个样式添加过渡效果，请添加多个属性，由逗号隔开。

例：向宽度、高度和转换添加过渡效果，代码如下。

```
div
{
transition: width 2s, height 2s, transform 2s;
-moz-transition: width 2s, height 2s, -moz-transform 2s;
-webkit-transition: width 2s, height 2s, -webkit-transform 2s;
-o-transition: width 2s, height 2s, -o-transform 2s;
}
```

表 5.9 中列出了所有的转换属性。

表 5.9　转换属性

属　　性	描　　述
transition	简写属性，用于在一个属性中设置 4 个过渡属性
transition-property	规定应用过渡的 CSS 属性的名称
transition-duration	定义过渡效果花费的时间，默认是 0
transition-timing-function	规定过渡效果的时间曲线，默认是 ease
transition-delay	规定过渡效果何时开始，默认是 0

下面的两个例子设置所有过渡属性。

例：以下代码中使用了所有过渡属性。

```
div
{
transition-property: width;
transition-duration: 1s;
transition-timing-function: linear;
transition-delay: 2s;
/* Firefox 4 */
-moz-transition-property:width;
-moz-transition-duration:1s;
-moz-transition-timing-function:linear;
-moz-transition-delay:2s;
/* Safari 和 Chrome */
-webkit-transition-property:width;
-webkit-transition-duration:1s;
-webkit-transition-timing-function:linear;
-webkit-transition-delay:2s;
/* Opera */
-o-transition-property:width;
-o-transition-duration:1s;
-o-transition-timing-function:linear;
-o-transition-delay:2s;
}
```

下面的代码与上面的例子具有相同的过渡效果，但是使用了简写的 transition 属性。

```
div
{
transition: width 1s linear 2s;
/* Firefox 4 */
-moz-transition:width 1s linear 2s;
/* Safari and Chrome */
-webkit-transition:width 1s linear 2s;
/* Opera */
-o-transition:width 1s linear 2s;
}
```

5.10 CSS3 渐变效果

渐变效果分为线性渐变和径向渐变效果。

(1) 线性渐变：左上(0% 0%)到右上(0% 100%)即从左到右水平渐变，以下的代码实现了左到右的渐变。

```
background-image: -webkit-gradient(linear,0% 0%,100% 0%,from(#2A8BBE),to(#FE280E));
```

这里 linear 表示线性渐变，从左到右，由蓝色（#2A8BBE）到红色（#FE280E）的渐变，效果如图 5.10 所示。

图 5.10　线性渐变

同理，也可以有从上到下，任何颜色间的渐变转换，效果如图 5.11 所示。

图 5.11　任何颜色间的渐变转换

还有复杂一点儿的渐变，如水平渐变，33%处为绿色，66%处为橙色。

```
background-image:-webkit-gradient(linear,0% 0%,100% 0%,from(#2A8BBE),
        color-stop(0.33,#AAD010),color-stop(0.33,#FF7F00),to(#FE280E));
```

这里的 color—stop 为拐点，效果如图 5.12 所示。

图 5.12　复杂的渐变

（2）径向渐变：径向渐变不是从一个点到一个点的渐变，而是从一个圆到一个圆的渐变。不是放射渐变而是径向渐变。来看一个例子，代码如下。

```
backgroud:
-webkit-gradient(radial,50 50,50,50 50,0,from(black),color-stop(0.5,red),to(blue));
```

前面“50，50，50”是起始圆的圆心坐标和半径，“50，50，0”是目标圆的圆心坐标和半径，“color-stop(0.5，red)”是断点的位置和色彩。这里需要说明一下，和放射由内至外不一样，径向渐变刚好相反，是由外到内的渐变。代码中标识的是两个同心圆，外圆半径为 50px，内圆半径为 0，那么就是从黑色到红色再到蓝色的正圆形渐变。如图 5.13 所示就是这段代码的效果。

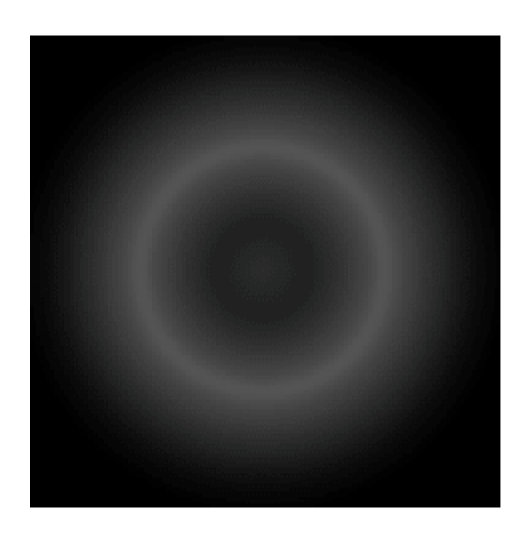


图 5.13　径向渐变

如果给目标源一个大于 0 的半径，将会看到另外一个效果。

```
backgroud:
-webkit-gradient(radial,50 50,50,50 50,10,from(black),color-stop(0.5,red),to(blue));
```

这里给目标圆半径为 10，效果如图 5.14 所示。

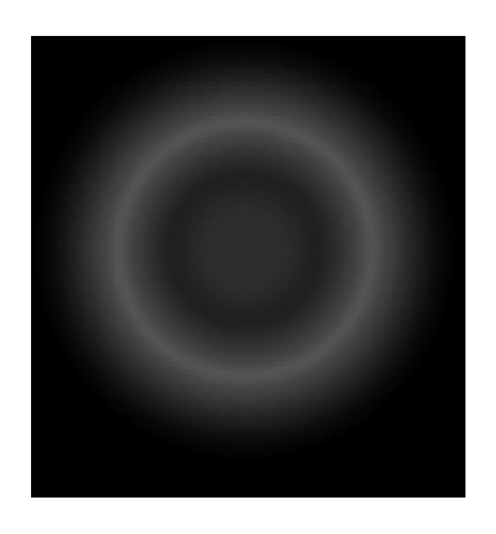


图 5.14　径向渐变

可以看到，会有一个半径为 10 的纯蓝的圆在最中间，这就是设置目标圆半径的效果。
现在再改变一下，不再是同心圆了，内圆圆心向右 20px 偏移。

```
backgroud:
-webkit-gradient(radial,50 50,50,70 50,10,from(black),color-stop(0.5,red),to(blue));
```

这里给目标圆半径还是 10，但是圆心偏移为“70，50”（起始圆圆心为“50，50”）效果如图 5.15 所示。

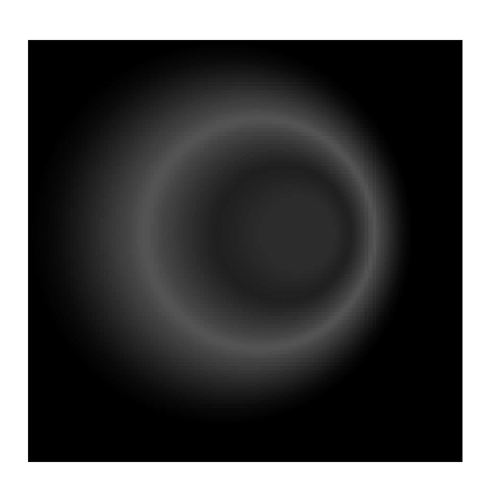


图 5.15　径向渐变

再来制作一个来自顶部的漫射光，类似于开了盏灯，代码如下，其效果如图 5.16 所示。

```
backgroud:
-webkit-gradient(radial,50 50,50,50 1,0,from(black),to(white));
```

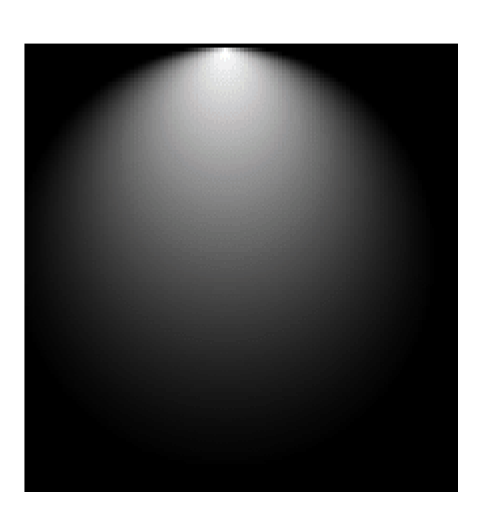


图 5.16　径向渐变

5.11　CSS3 反射效果

反射的效果就像水中的倒影，其设置也很简单，参考如下代码。

```
.classReflect{
-webkit-box-reflect: below 10px
-webkit-gradient(linear, left top, left bottom, from(transparent),
      to(rgba(255, 255, 255, 0.51)));
}
```

主要关注“-webkit-box-reflect：below 10px”，它表示反射在元素下方 10px 的地方，再配上渐变效果，效果图如图 5.17 所示。

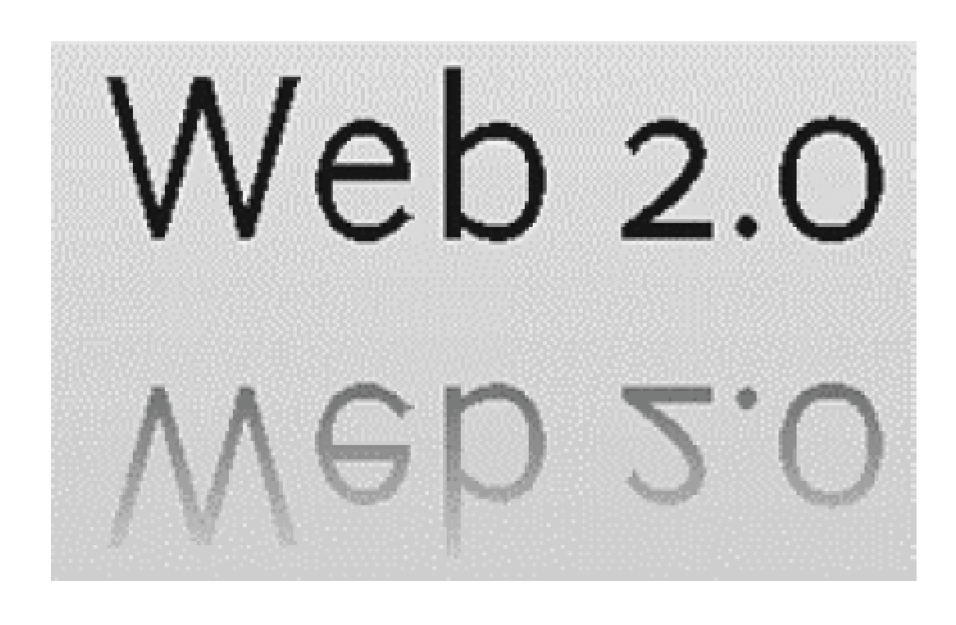


图 5.17　反射效果

5.12　CSS3 动画

通过 CSS3 可以创建动画，这可以在许多网页中取代动画图片、Flash 动画以及 JavaScript。如需在 CSS3 中创建动画，需要学习@keyframes 规则。

@keyframes 规则用于创建动画。在@keyframes 中规定某项 CSS 样式，就能创建由当前样式逐渐改为新样式的动画效果。

IE 仍然不支持@keyframes 规则或动画属性。Firefox 需要加前缀-moz-。Chrome 和 Safari 需要加前缀-webkit-。Opera 需要加前缀-o-。

例：

```
@keyframes myfirst
{
from {background: red;}
to {background: yellow;}
}

@-moz-keyframes myfirst /* Firefox */
{
from {background: red;}
```

```
to {background: yellow;}
}

@-webkit-keyframes myfirst /* Safari 和 Chrome */
{
from {background: red;}
to {background: yellow;}
}

@-o-keyframes myfirst /* Opera */
{
from {background: red;}
to {background: yellow;}
}
```

当在@keyframes 中创建动画时，应把它捆绑到某个选择器，否则不会产生动画效果。

通过规定至少以下两项 CSS3 动画属性，即可将动画绑定到选择器。

(1) 规定动画的名称。

(2) 规定动画的时长。

例：把 myfirst 动画捆绑到 div 元素，时长 5s。

```
div
{
animation: myfirst 5s;
-moz-animation: myfirst 5s;      /* Firefox */
-webkit-animation: myfirst 5s;   /* Safari 和 Chrome */
-o-animation: myfirst 5s;        /* Opera */
}
```

注释：必须定义动画的名称和时长。如果忽略时长，则动画不会允许，因为默认值是 0。

可以使用百分比来规定变化发生的时间，或用关键词"from"和"to"，等同于 0%和 100%。0%是动画的开始，100%是动画的完成。

为了得到最佳的浏览器支持，应该始终定义 0%和 100%选择器。

例：当动画 25%及 50%时改变背景色，然后当动画 100%完成时再次改变。

```
@keyframes myfirst
{
0%   {background: red;}
25%  {background: yellow;}
50%  {background: blue;}
100% {background: green;}
}

@-moz-keyframes myfirst /* Firefox */
{
0%   {background: red;}
```

```
25 %  {background: yellow;}
50 %  {background: blue;}
100 % {background: green;}
}

@ - webkit - keyframes myfirst / *  Safari 和 Chrome  * /
{
0 %   {background: red;}
25 %  {background: yellow;}
50 %  {background: blue;}
100 % {background: green;}
}

@ - o - keyframes myfirst / *  Opera  * /
{
0 %   {background: red;}
25 %  {background: yellow;}
50 %  {background: blue;}
100 % {background: green;}
}
```

例：改变背景色和位置。

```
@keyframes myfirst
{
0 %    {background: red; left:0px; top:0px;}
25 %   {background: yellow; left:200px; top:0px;}
50 %   {background: blue; left:200px; top:200px;}
75 %   {background: green; left:0px; top:200px;}
100 %  {background: red; left:0px; top:0px;}
}

@ - moz - keyframes myfirst / *  Firefox  * /
{
0 %    {background: red; left:0px; top:0px;}
25 %   {background: yellow; left:200px; top:0px;}
50 %   {background: blue; left:200px; top:200px;}
75 %   {background: green; left:0px; top:200px;}
100 %  {background: red; left:0px; top:0px;}
}

@ - webkit - keyframes myfirst / *  Safari 和 Chrome  * /
{
0 %    {background: red; left:0px; top:0px;}
25 %   {background: yellow; left:200px; top:0px;}
50 %   {background: blue; left:200px; top:200px;}
75 %   {background: green; left:0px; top:200px;}
100 %  {background: red; left:0px; top:0px;}
}
```

```
@-o-keyframes myfirst /* Opera */
{
0%   {background: red; left:0px; top:0px;}
25%  {background: yellow; left:200px; top:0px;}
50%  {background: blue; left:200px; top:200px;}
75%  {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}
```

表 5.10 中列出了@keyframes 规则和所有动画属性。

表 5.10　@keyframes 规则和所有动画属性

属　　性	描　　述
@keyframes	规定动画
animation	所有动画属性的简写属性，除了 animation-play-state 属性
animation-name	规定@keyframes 动画的名称
animation-duration	规定动画完成一个周期所花费的时间(秒或毫秒)，默认是 0
animation-timing-function	规定动画的速度曲线。默认是 ease
animation-delay	规定动画何时开始。默认是 0
animation-iteration-count	规定动画被播放的次数。默认是 1
animation-direction	规定动画是否在下一周期逆向地播放。默认是 normal
animation-play-state	规定动画是否正在运行或暂停。默认是 running

下面的两个例子设置了所有动画属性。

例：设置所有动画属性。

```
div
{
animation-name: myfirst;
animation-duration: 5s;
animation-timing-function: linear;
animation-delay: 2s;
animation-iteration-count: infinite;
animation-direction: alternate;
animation-play-state: running;
/* Firefox: */
-moz-animation-name: myfirst;
-moz-animation-duration: 5s;
-moz-animation-timing-function: linear;
-moz-animation-delay: 2s;
-moz-animation-iteration-count: infinite;
-moz-animation-direction: alternate;
-moz-animation-play-state: running;
/* Safari 和 Chrome: */
-webkit-animation-name: myfirst;
```

```
-webkit-animation-duration: 5s;
-webkit-animation-timing-function: linear;
-webkit-animation-delay: 2s;
-webkit-animation-iteration-count: infinite;
-webkit-animation-direction: alternate;
-webkit-animation-play-state: running;
/* Opera: */
-o-animation-name: myfirst;
-o-animation-duration: 5s;
-o-animation-timing-function: linear;
-o-animation-delay: 2s;
-o-animation-iteration-count: infinite;
-o-animation-direction: alternate;
-o-animation-play-state: running;
}
```

例：与上面的动画相同,但是使用了简写的动画 animation 属性。

```
div
{
animation: myfirst 5s linear 2s infinite alternate;
/* Firefox: */
-moz-animation: myfirst 5s linear 2s infinite alternate;
/* Safari 和 Chrome: */
-webkit-animation: myfirst 5s linear 2s infinite alternate;
/* Opera: */
-o-animation: myfirst 5s linear 2s infinite alternate;
}
```

5.13 CSS3 多列

通过 CSS3,可以创建多个列来对文本进行布局,在本节中将学习如下多列属性:column-count、column-gap 和 column-rule。

IE 不支持多列属性。Firefox 需要加前缀-moz-。Chrome 和 Safari 需要加前缀-webkit-。

column-count 属性规定元素应该被分隔的列数。

例：把 div 元素中的文本分隔为三列。

```
div
{
-moz-column-count:3;          /* Firefox */
-webkit-column-count:3;       /* Safari 和 Chrome */
column-count:3;
}
```

column-gap 属性规定列之间的间隔。

例：规定列之间 40px 的间隔。

```
div
{
-moz-column-gap:40px;          /* Firefox */
-webkit-column-gap:40px;       /* Safari 和 Chrome */
column-gap:40px;
}
```

column-rule 属性设置列之间的宽度、样式和颜色规则。

例：规定列之间的宽度、样式和颜色规则。

```
div
{
-moz-column-rule:3px outset #ff0000;          /* Firefox */
-webkit-column-rule:3px outset #ff0000;       /* Safari 和 Chrome */
column-rule:3px outset #ff0000;
}
```

表 5.11 中列出了所有的转换属性。

表 5.11　转换属性

属　性	描　述
column-count	规定元素应该被分隔的列数
column-fill	规定如何填充列
column-gap	规定列之间的间隔
column-rule	设置所有 column-rule-* 属性的简写属性
column-rule-color	规定列之间规则的颜色
column-rule-style	规定列之间规则的样式
column-rule-width	规定列之间规则的宽度
column-span	规定元素应横跨的列数
column-width	规定列的宽度
columns	规定设置 column-width 和 column-count 的简写属性

5.14　CSS3 用户界面

在 CSS3 中，新的用户界面特性包括重设元素尺寸、盒尺寸以及轮廓等。在本节中，将学到以下用户界面属性：resize、box-sizing 和 outline-offset。

Firefox 4+、Chrome 以及 Safari 支持 resize 属性。IE，Chrome，Opera 支持 box-sizing 属性。Firefox 需要加前缀-moz-。Safari 需要加前缀-webkit-。所有主流浏览器都支持 outline-offset 属性，除了 IE。

resize 属性规定是否可由用户调整元素尺寸。

例：规定 div 元素可由用户调整大小。

```
div
{
resize:both;
overflow:auto;
}
```

box-sizing 属性允许以确切的方式定义适应某个区域的具体内容。

例：规定两个并排的带边框方框。

```
div
{
box-sizing:border-box;
-moz-box-sizing:border-box;          /* Firefox */
-webkit-box-sizing:border-box;       /* Safari */
width:50%;
float:left;
}
```

outline-offset 属性对轮廓进行偏移，并在超出边框边缘的位置绘制轮廓。

轮廓与边框有以下两点不同。

(1) 轮廓不占用空间。

(2) 轮廓可能是非矩形。

例：规定边框边缘之外 15px 处的轮廓。

```
div
{
border:2px solid black;
outline:2px solid red;
outline-offset:15px;
}
```

表 5.12 中列出了所有的转换属性。

表 5.12　转换属性

属　　性	描　　述
appearance	允许将元素设置为标准用户界面元素的外观
box-sizing	允许以确切的方式定义适应某个区域的具体内容
icon	为创作者提供使用图标化等价物来设置元素样式的能力
nav-down	规定在使用 arrow-down 导航键时向何处导航
nav-index	设置元素的 Tab 键控制次序
nav-left	规定在使用 arrow-left 导航键时向何处导航
nav-right	规定在使用 arrow-right 导航键时向何处导航
nav-up	规定在使用 arrow-up 导航键时向何处导航
outline-offset	对轮廓进行偏移，并在超出边框边缘的位置绘制轮廓
resize	规定是否可由用户对元素的尺寸进行调整

5.15 Media Queries

Media Queries 翻译过来就是“媒体查询”的意思，Web 页面中 head 部分常看到这样的一段代码：

```
<link href="css/reset.css" rel="stylesheet" type="text/css" media="screen" />
<link href="css/style.css" rel="stylesheet" type="text/css" media="all" />
<link href="css/print.css" rel="stylesheet" type="text/css" media="print" />
```

或者这样的形式：

```
<style type="text/css" media="screen">
@import url("css/style.css");
</style>
```

两种方式引入 CSS 样式都有一个共同的属性 media，而这个 media 就是用来指定特定的媒体类型，在 HTML4 和 CSS2 中允许使用 media 来指定特定的媒体类型，如屏幕(screen)和打印(print)的样式表等。

CSS3 中的 Media Queries 增加了更多的媒体查询，同时可以添加不同的媒体类型的表达式用来检查媒体是否符合某些条件，如果媒体符合相应的条件，那么就会调用对应的样式表。简单地说，“在 CSS3 中可以设置不同类型的媒体条件，并根据对应的条件，给相应符合条件的媒体调用相对应的样式表”。

现在最常见的一个例子，用户可以同时给 PC 的大屏幕和移动设备设置不同的样式表。这个功能是非常强大的，它可以让用户定制不同的分辨率和设备，并在不改变内容的情况下，让用户制作的 Web 页面在不同的分辨率和设备下都能显示正常，并且不会因此而丢失样式。

首先来看一个简单的实例：

```
<link rel="stylesheet" media="screen and (max-width: 600px)" href="small.css" />
```

上面的 media 语句表示的是：当页宽度小于或等于 600px 时，调用 small.css 样式表来渲染 Web 页面。首先来看 media 的语句中包含的内容。

(1) screen：指的是一种媒体类型。

(2) and：被称为关键词，与其相似的还有 not，only。

(3) (max-width:600px)：这个就是媒体特性，说得通俗一点儿就是媒体条件。

前面这个简单的实例引出两个概念性的东西，媒体类型(Media Type)和媒体特性(Media Query)，首先一起来理解一下这两个概念。

1. 媒体类型

媒体类型(Media Type)在 CSS2 中是一个常见的属性，也是一个非常有用的属性，可以通过媒体类型对不同的设备指定不同的样式，在 CSS2 中常碰到的就是 all(全部)，screen(屏幕)，print(页面打印或打印预览模式)。其实媒体类型不止这三种，W3C 总共列出了 10

种媒体类型。页面中引入媒体类型的方法也有多种，如下。

1）link 方法引入

```
<link rel = "stylesheet" type = "text/css" href = "../css/print.css" media = "print" />
```

2）xml 方式引入

```
<?xml - stylesheet rel = "stylesheet" media = "screen" href = "css/style.css"?>
```

3）@import 方式引入

@import 引入有两种方式，一种是在样式文件中通过@import 调用别一个样式文件；另一种方法是在<head></head>中的<style>…</style>中引入，但这种使用方法在 IE 6 和 IE 7 中都不被支持，如一个样式文件中调用另一个样式文件：

```
@import url("css/reset.css") screen;
@import url("css/print.css") print;
```

在<head></head>中的<style>…</style>中调用：

```
<head>
  <style type = "text/css">
  @import url("css/style.css") all;
  </style>
</head>
```

4）@media 引入

这种引入方式和@import 是一样的，也有以下两种方式。

(1) 样式文件中使用：

```
@media screen{
  选择器{
 属性: 属性值;
  }
}
```

(2) 在<head></head>中的<style>…</style>中调用：

```
<head>
  <style type = "text/css">
  @media screen{
         选择器{
       属性: 属性值;
     }
  }
  </style>
</head>
```

以上几种方法都有其各自的利弊，在实际应用中建议使用第一种和第四种，因为这两种方法是在项目制作中常用的方法。

2. 媒体特性

前面有简单的提到，Media Query 是 CSS3 对 Media Type 的增强版，其实可以将 Media Query 看成 Media Type(判断条件)＋CSS(符合条件的样式规则)，常用的特性 W3C 共列出来 13 种，具体可以参阅 Media Features。为了深刻理解 Media Query，再次回到前面的实例上：

```
<link rel = "stylesheet" media = "screen and (max - width: 600px)" href = "small.css" />
```

转换成 CSS 中的写法为：

```
@media screen and (max - width: 600px) {
    选择器 {
      属性: 属性值;
    }
  }
```

其实就是把 small.css 文件中的样式放在了@media srceen and (max-width;600px){…}的大括号之中。在语句上面的语句结构中，可以看出 Media Query 和 CSS 的属性集合很相似，主要区别有以下几点。

(1) Media Query 只接受单个的逻辑表达式作为其值，或者没有值；

(2) CSS 属性用于声明如何表现页面上的信息；而 Media Query 是一个用于判断输出设备是否满足某种条件的表达式；

(3) Media Query 其中的大部分接受 min/max 前缀，用来表示其逻辑关系，表示应用于大于等于或者小于等于某个值的情况；

(4) CSS 属性要求必须有属性值，Media Query 可以没有值，因为其表达式返回的只有真或假两种。

下面来看看 Media Queries 的具体使用方式。

1. 最大宽度 Max Width

```
<link rel = "stylesheet" media = "screen and (max - width:600px)" href = "small.css" type =
"text/css" />
```

上面表示的是：当屏幕小于或等于 600px 时，将采用 small.css 样式来渲染 Web 页面。

2. 最小宽度 Min Width

```
<link rel = "stylesheet" media = "screen and (min - width:900px)" href = "big.css" type = "text/
css"/>
```

上面表示的是：当屏幕大于或等于 900px 时，将采用 big.css 样式来渲染 Web 页面。

3. 多个 Media Queries 使用

```
<link rel = "stylesheet" media = "screen and (min - width:600px) and (max - width:900px)" href
 = "style.css" type = "text/css"/>
```

Media Query 可以结合多个媒体查询，换句话说，一个 Media Query 可以包含 0 到多个表达式，表达式又可以包含 0 到多个关键字，以及一种 Media Type。正如上面的代码表示的是当屏幕在 600～900px 之间时采用 style.css 样式来渲染 Web 页面。

4. 设备屏幕的输出宽度 Device Width

```
<link rel = "stylesheet" media = "screen and (max - device - width: 480px)"
href = "iphone.css" type = "text/css" />
```

上面的代码指的是 iphone.css 样式适用于最大设备宽度为 480px，例如 iPhone 上的显示，这里的 max-device-width 所指的是设备的实际分辨率，也就是指可视面积分辨率。

5. iPhone4

```
<link rel = "stylesheet" media = "only screen and ( - webkit - min - device - pixel - ratio: 2)"
type = "text/css" href = "iphone4.css" />
```

上面的样式是专门针对 iPhone4 的移动设备写的。

6. iPad

```
<link rel = "stylesheet" media = "all and (orientation:portrait)"
href = "portrait.css" type = "text/css" />
<link rel = "stylesheet" media = "all and (orientation:landscape)"
href = "landscape.css"   type = "text/css" />
```

在大数情况下，移动设备 iPad 上的 Safari 和在 iPhone 上的是相同的，只是 iPad 声明了不同的方向，如上例，在纵向(portrait)时采用 portrait.css 来渲染页面；在横向(landscape)时采用 landscape.css 来渲染页面。

7. Android

```
/* 240px 的宽度 */
<link rel = "stylesheet" media = "only screen and (max - device - width: 240px)" href =
"android240.css" type = "text/css" />
/* 360px 的宽度 */
<link rel = "stylesheet" media = "only screen and (min - device - width:241px) and (max - device
 - width:360px)" href = "android360.css" type = "text/css" />
/* 480px 的宽度 */
<link rel = "stylesheet" media = "only screen and (min - device - width:361px) and (max - device
 - width:480px)" href = "android480.css" type = "text/css" />
```

可以使用 Media Query 为 Android 手机在不同分辨率提供特定样式，这样就可以解决屏幕分辨率的不同给 Android 手机的页面带来的重构问题。

8. not 关键字

```
<link rel="stylesheet" media="not print and (max-width: 1200px)" href="print.css" type
="text/css" />
```

not 关键字是用来排除某种制定的媒体类型，换句话说就是用于排除符合表达式的设备。

9. only 关键字

```
<link rel="stylesheet" media="only screen and (max-device-width:240px)" href=
"android240.css" type="text/css" />
```

only 用来定某种特定的媒体类型，可以用来排除不支持媒体查询的浏览器。其实 only 很多时候是用来对那些不支持 Media Query 但却支持 Media Type 的设备隐藏样式表的。

其主要有：支持媒体特性（Media Queries）的设备，正常调用样式，此时就当 only 不存在；对于不支持媒体特性（Media Queries）但又支持媒体类型（Media Type）的设备，这样就会不读样式，因为其先读 only 而不是 screen；另外不支持 Media Queries 的浏览器，不论是否支持 only，样式都不会被采用。

10. 其他

在 Media Query 中如果没有明确指定 Media Type，那么其默认为 all，如：

```
<link rel="stylesheet" media="(min-width: 701px) and (max-width: 900px)" href="medium.
css" type="text/css"/>
```

另外还有使用逗号（,）用来表示并列或者表示或，如：

```
<link rel="stylesheet" type="text/css" href="style.css" media="handheld and (max-
width:480px), screen and (min-width:960px)"/>
```

上面代码中，style.css 样式被用在宽度小于或等于 480px 的手持设备上，或者被用于屏幕宽度大于或等于 960px 的设备上。

关于 Media Query 的使用这一节就介绍到此，最后总体归纳一下其功能：Media Queries 能在不同的条件下使用不同的样式，使页面达到不同的渲染效果。

小　　结

在本章中，主要介绍了 CSS3 在 HTML5 移动 Web 开发中的功能。通过学习 CSS3 的特性，可以在移动 Web 开发中对页面布局、字体、颜色、背景等方面实现有效的控制。

第6章 jQuery Mobile

本章学习目标

- jQuery Mobile
- jQuery Mobile 基本页面结构
- jQuery Mobile 基本元素

jQuery 一直以来都是非常流行的客户端及 Web 应用程序开发中使用的 JavaScript 类库，然而一直以来它都是为桌面浏览器设计的，没有特别为移动应用程序设计。

jQuery Mobile 是一个新的项目，用来添补在移动设备应用上的缺憾。它是基本 jQuery 框架并提供了一定范围的用户接口和特性，以便于开发人员在移动应用上使用。使用该框架可以节省大量的 JavaScript 代码开发时间，尽管目前的版本还不是一个稳定的版本，但它的应用效果已经备受瞩目。

jQuery Mobile 为开发移动应用程序替代了非常简单的用户接口。这种接口的配置是标签驱动的，这意味着可以在 HTML 中建立大量的程序接口而不需要写一行 JavaScript 代码。

jQuery Mobile 提供了一些自定义的事件用来探测移动和触摸动作，例如 tap（敲击）、tap-and-hold（点击并按住）、swipe、orientation change。

jQuery Mobile 基本特性如下。

（1）一般简单性和灵活性。

该框架易于使用。主要使用标记驱动开发页面，无须或仅需很少的 JavaScript；使用高级 JavaScript 和事件；使用一个 HTML 文件和多个嵌入页面；将应用程序分解成多个页面。

（2）逐步强化和全面兼容。

尽管 jQuery Mobile 利用最新的 HTML5、CSS3 和 JavaScript，但并非所有移动设备都提供这样的支持。jQuery Mobile 的理念是同时支持高端和低端设备，例如那些不支持 JavaScript 的设备，尽量提供最好的体验。

（3）支持触摸屏输入和其他输入方法。

jQuery Mobile 为不同输入方法和事件提供支持：触摸屏、鼠标和基于光标焦点的用户输入。

（4）可访问性。

jQuery Mobile 在设计时考虑了访问能力，它支持 Accessible Rich Internet Applications（WAI-ARIA），以帮助使用辅助技术的残障人士访问 Web 页面。

（5）轻量级和模块化。

该框架属于轻量级，拥有一个很小的 JavaScript 库、CSS 以及一些图标。

（6）主题。

该框架还提供一个主题系统，允许定义自己的应用程序样式，可以轻松地创建自己的主题。

jQuery Mobile 框架包括构建完整移动 Web 应用程序和网站所需的所有 UI 组件，如页面、对话框、工具栏、不同类型的列表视图、各种表单元素和按钮等。jQuery Mobile 构建于 jQuery 内核之上，因此可以访问关键设备，包括 HTML 和 XML 文档对象模型(DOM)的遍历操作；事件处理；使用 AJAX 服务器通信；以及 Web 页面的动画和图像效果。有了 jQuery Mobile，就可以轻而易举地编写基础 Web 应用程序。由于 jQuery Mobile 是一个非常全面的基础架构，提供了一些高级事件和 API，所以还可以编写高级移动 Web 应用程序和网站。

接下来将通过实例展示 jQuery Mobile 的特性及好处，可以看到这个新框架是如何在短时间内建立起一个高质量的移动应用程序，讲解的代码最好使用的移动设备平台是 iPhone 或 Android。或者是在 PC 上使用 Safari 浏览器调试。

6.1 浏览器支持

移动设备浏览器经历了漫长的发展，但并非所有移动设备都支持 HTML5、CSS3 和 JavaScript。这个领域是 jQuery Mobile 持续增强和支持全面兼容发挥作用的地方。jQuery Mobile 同时支持高端和低端设备，包括那些不支持 JavaScript 的设备。持续增强是一个设计理念，包含以下核心原则。

（1）所有浏览器都应该能够访问全部基础内容。

（2）所有浏览器都应该能够访问全部基础功能。

（3）增强的布局由外部链接的 CSS 提供。

（4）增强的行为由外部链接的 CSS 提供。

（5）终端用户浏览器偏好应受到重视。

所有基本内容应该在基础设备上进行渲染，而更高级的平台和浏览器将使用额外的、外部链接的 JavaScript 和 CSS 持续增强。

jQuery Mobile 为大量移动设备提供支持，jQuery Mobile 将设备支持根据其支持级别分类或分组成三个类别。

A 级：设备支持一个充分增强的用户体验和基于 Ajax 的动画页面过渡。jQuery Mobile 支持超过 20 个不同设备，其中包括：iOS 3.2～5.0、Android 2.1～2.3 与 3.0、BlackBerry 6 和 7 与 Playbook、Skyfire4.1、Opera Mobile；还支持桌面浏览器 Chrome、Firefox、IE 和 Opera。

B 级：设备支持一个增强体验，但是没有 Ajax 导航特性。支持的设备包括：BlackBerry 5.0、Opera Mini 5.0～6.5 和 Nokia Symbian。

C 级：设备支持一个基本的、非增强的 HTML 体验。支持的设备包括：上一代智能手机，包括 BlackBerry 4.x、Windows Mobile 和其他设备。

6.2 jQuery Mobile 基本页面结构

一个典型的 jQuery Mobile 页面有三部分：页眉、内容和页脚部分。页眉部分通常放置页面标题和 logo 等信息，内容部分放置特定 Web 应用程序和各种小部件，页脚部分很适合导航。

要利用 jQuery Mobile 高级功能和 HTML 语法，jQuery Mobile HTML 文档必须定义 HTML5 文档类型。HTML 文档其余部分包括<head>和<body>两部分，其中，<head>包含 jQuery Mobile 引用和 CSS 移动主题，<body>部分包含 jQuery Mobile 内容。大部分 jQuery Mobile Web 应用程序都要遵循下面的基本模板。

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
    <link rel = "stylesheet" href = "http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.2.min.css">
    <script src = "http://code.jquery.com/jquery-1.8.3.min.js"></script>
    <script src = "http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.2.min.js"></script>
</head>
<body>
    <div data-role = "page">
    <div data-role = "header">  <h1>Page Title</h1></div>
    <div data-role = "content">
     <p>Page content goes here.</p>
    </div>
  <div data-role = "footer">
  <h4>Page Footer</h4>
  </div>
  </div>
 </body>
</html>
```

要使用 jQuery Mobile，首先需要在开发的界面中包含 jQuery.mobile-1.3.2.min.css、jQuery-1.8.3.min.js 和 jQuery.mobile-1.3.2.min.js 这三个文件，如果没有，开发人员也可以去网上下载它们。

在模板中有些地方值得注意。首先是 DIV 元素的使用，既然 HTML5 在移动设备中如此流行，为什么不使用更加新的 header、article、section 和 footer 元素呢？这是因为较老的智能手机浏览器无法明白新的 HTML5 元素。在某些情况下，例如，Windows Phone 上老版本的 IE 会出现一个 Bug 使得无法加载页面的 CSS。而 DIV 元素却被广泛支持。

此时已经可以保存网页并在浏览器中查看了，这些代码同样可以在桌面浏览器中正常工作。推荐使用 Chrome 或者 Opera Mobile Emulator 来进行本地测试。要在真实环境测试，就需要相应的移动设备了。基本模板显示效果如图 6.1 所示。

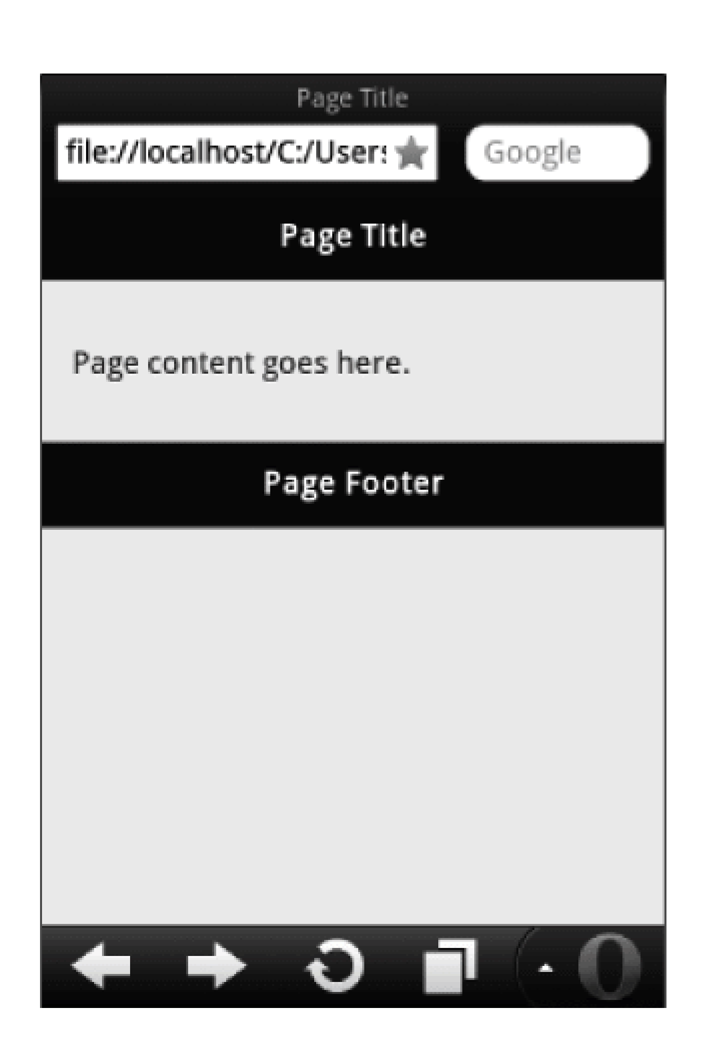


图 6.1　基本模板显示效果

可以看到页面中的内容都是包装在 div 标签中并在标签中加入 data-role＝"page"属性。这样 jQuery Mobile 就会知道哪些内容需要处理。

说明：data-属性是 HTML5 新推出的一个很有趣的特性，它可以让开发人员添加任意属性到 HTML 标签中，只要添加的属性名有“data-”前缀。

在“page”div 中，还可以包含“header”“content”“footer”的 div 元素，这些元素都是可选的，但至少要包含一个“content”div，各元素的作用如表 6.1。

表 6.1　各元素的作用

<div data-role＝"header"></div>	在页面的顶部建立导航工具栏，用于放置标题和按钮(典型的至少要放一个“返回”按钮，用于返回前一页)。通过添加额外的属性 data-position＝"fixed"，可以保证头部始终保持在屏幕的顶部
<div data-role＝"content"></div>	包含一些主要内容，例如文本内容、图像、按钮、列表、表单等
<div data-role＝"footer"></div>	在页面的底部建立工具栏，添加一些功能按钮 为了确保它始终保持在页面的底部，可以给其加上 data-position＝"fixed"属性

另外，关于单个页面、多个页面和页面链接问题，这里也要说一下。单个 HTML 文档可以有一个或多个 jQuery Mobile 页面，正如下面的代码中的 data-role＝"page"属性所显示的。如果有多个页面，必须为 id 属性指定一个唯一的页面 ID，用于链接外部页面。加载一个多页面 HTML 文档时，仅显示第一个页面。

```
<body>
<div data-role="page" id="page1">
    <div data-role="header">
        <h1>HEADER</h1>
    </div>
```

```
        <div data-role="content">
            <p>
            CONTENT AREA
            </p>
        </div>
        <div data-role="footer">
            <h1>FOOTER</h1>
        </div>
    </div>

    <div data-role="page" id="page2">
        <div data-role="header">
            <h1>HEADER</h1>
        </div>
        <div data-role="content">
            <p>
            CONTENT AREA 2
            </p>
        </div>
        <div data-role="footer">
            <h1>FOOTER</h1>
        </div>
    </div>
    </body>
```

6.3 jQuery Mobile 过渡

jQuery Mobile 包含选择页面打开方式的 CSS 效果。jQuery Mobile 拥有一系列关于如何从一页过渡到下一页的效果。如需实现过渡效果,浏览器必须支持 CSS3 3D 转换。IE 10 支持 3D 转换(更早的版本不支持),但 Opera 仍然不支持 3D 转换。

过渡效果可应用于任意链接或通过使用 data-transition 属性进行的表单提交,如以下示例代码所示。

```
<a href="#anylink" data-transition="slide">滑动到页面二</a>
```

表 6.2 展示了可与 data-transition 属性一同使用的可用过渡。

表 6.2 可用的过渡

过　渡	描　述	过　渡	描　述
fade	默认。淡入淡出到下一页	slidefade	从右向左滑动并淡入到下一页
flip	从后向前翻动到下一页	slideup	从下到上滑动到下一页
flow	抛出当前页面,引入下一页	slidedown	从上到下滑动到下一页
pop	像弹出窗口那样转到下一页	turn	转向下一页
slide	从右向左滑动到下一页	none	无过渡效果

提示：在 jQuery Mobile 中，淡入淡出效果在所有链接上都是默认的（如果浏览器支持）。以上所有效果同时支持反向动作，例如，如果希望页面从左向右滑动，而不是从右向左，请使用值为 reverse 的 data-direction 属性。在"后退"按钮上是默认的。

下面的代码是从右向左滑动以及从左向右滑动的例子。

```
<!DOCTYPE html>
<html>
<head>
<link rel = "stylesheet"
 href = "http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.2.min.css">
<script src = "http://code.jquery.com/jquery-1.8.3.min.js"></script>
<script src = "http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.2.min.js"></script>
</head>
<body>

<div data-role = "page" id = "pageone">
  <div data-role = "header">
    <h1>欢迎来到我的主页</h1>
  </div>

  <div data-role = "content">
    <p>点击链接来查看滑动效果(从右向左滑动到下一页)</p>
    <a href = "#pagetwo" data-transition = "slide">滑动到页面二</a>
  </div>

  <div data-role = "footer">
  <h1>页脚文本</h1>
  </div>
</div>

<div data-role = "page" id = "pagetwo">
  <div data-role = "header">
    <h1>欢迎来到我的主页</h1>
  </div>

  <div data-role = "content">
    <p>点击链接来查看反向的滑动效果(从左向右滑动到前一页)</p>
    <a href = "#pageone" data-transition = "slide" data-direction = "reverse">
     滑动到页面一(反向)
     </a>
  </div>

  <div data-role = "footer">
  <h1>页脚文本</h1>
  </div>
</div>

</body>
</html>
```

6.4 jQuery Mobile 按钮

在 jQuery Mobile 中的按钮可以通过以下三种方法创建。

(1) 使用<button>元素；

(2) 使用<input>元素；

(3) 使用 data-role="button"的<a>元素。

先看看<button>的用法，格式如下。

```
<button>按钮</button>
```

再来看<input>的用法，格式如下。

```
<input type = "button" value = "按钮">
```

最后来看<a>的用法，格式如下。

```
<a href = "#" data-role = "button">按钮</a>
```

注意：jQuery Mobile 中的按钮会自动获得样式，这增强了它们在移动设备上的交互性和可用性。推荐使用 data-role="button"的<a>元素来创建页面之间的链接，而<input>或<button>元素用于表单提交。

导航按钮：如需通过按钮来链接页面，请使用 data-role="button"的<a>元素，如下面的代码。

```
<a href = "#pagetwo" data-role = "button">转到页面二</a>
```

行内按钮：默认情况下，按钮会占据屏幕的全部宽度。如果需要按钮适应其内容，或者如果需要两个或多个按钮并排显示，请添加 data-inline="true"，如下面的代码。

```
<a href = "#pagetwo" data-role = "button" data-inline = "true">转到页面二</a>
```

组合按钮：jQuery Mobile 提供了对按钮进行组合的简单方法。可以同时使用两个属性，也就是 data-role="controlgroup"属性与 data-type="horizontal|vertical"一同使用，以规定水平或垂直地组合按钮，如下面的代码。

```
<div data-role = "content">
    <div data-role = "controlgroup" data-type = "horizontal">
    <p>水平分组：</p>
    <a href = "#" data-role = "button">按钮 1</a>
    <a href = "#" data-role = "button">按钮 2</a>
    <a href = "#" data-role = "button">按钮 3</a>
    </div><br>

    <div data-role = "controlgroup" data-type = "vertical">
```

```
        <p>垂直分组(默认): </p>
        <a href = "#" data - role = "button">按钮 1 </a>
        <a href = "#" data - role = "button">按钮 2 </a>
        <a href = "#" data - role = "button">按钮 3 </a>
        </div>
</div>
```

提示：默认情况下，组合按钮是垂直分组的，彼此间没有外边距和空白。并且只有第一个和最后一个按钮拥有圆角，在组合后就创造出了漂亮的外观。

后退按钮：如需创建后退按钮，请使用 data-rel＝"back"属性(会忽略 href 值)，如下面的代码。

```
<a href = "#" data - role = "button" data - rel = "back">返回</a>
```

更多用于按钮的 data-＊属性，如表 6.3 所示。

表 6.3　按钮的 data-＊属性

属　　性	值	描　　述
data-corners	true | false	规定按钮是否有圆角
data-mini	true | false	规定是否是小型按钮
data-shadow	true | false	规定按钮是否有阴影

6.5　jQuery Mobile 按钮图标

jQuery Mobile 提供的一套图标可令按钮更具吸引力，如需向按钮添加图标，请使用 data-icon 属性，如以下的代码。

```
<a href = "#anylink" data - role = "button" data - icon = "search">搜索</a>
```

提示：也可以在<button>或<input>元素中使用 data-icon 属性，下面列出了 jQuery Mobile 提供的一些可用图标，如表 6.4 所示。

表 6.4　jQuery Mobile 提供的图标

属　性　值	描　　述	图　　标
data-icon＝"arrow-l"	左箭头	
data-icon＝"arrow-r"	右箭头	
data-icon＝"delete"	删除	
data-icon＝"info"	信息	
data-icon＝"home"	首页	
data-icon＝"back"	返回	
data-icon＝"search"	搜索	
data-icon＝"grid"	网格	

定位图标：能够规定图标被放置的位置——上、右、下或左。请使用 data-iconpos 属性来规定位置，默认所有按钮中的图标靠左放置，如下面的代码。

```
<a href = "#link" data-role = "button" data-icon = "search" data-iconpos = "top">上</a>
<a href = "#link" data-role = "button" data-icon = "search" data-iconpos = "right">右</a>
<a href = "#link" data-role = "button" data-icon = "search" data-iconpos = "bottom">下</a>
<a href = "#link" data-role = "button" data-icon = "search" data-iconpos = "left">左</a>
```

只显示图标：如果只需显示图标，请将 data-iconpos 设置为 notext，如下面的代码。

```
<a href = "#link" data-role = "button" data-icon = "search" data-iconpos = "notext">搜索</a>
```

6.6 jQuery Mobile 工具栏

工具栏元素常被放置于页眉和页脚中，以实现“已访问”的导航。

1. 标题栏

页眉通常会包含页眉标题或一到两个按钮(通常是首页、选项或“搜索”按钮)。可以在页眉中向左侧或/以及右侧添加按钮。如下面的代码，将向页眉标题文本的左侧添加一个按钮，并向右侧添加一个按钮，显示效果如图 6.2 所示。

```
<div data-role = "header">
  <a href = "#" data-role = "button">首页</a>
  <h1>欢迎来到我的主页</h1>
  <a href = "#" data-role = "button">搜索</a>
</div>
```



图 6.2　标题

下面的代码将向页眉标题的左侧添加一个按钮。

```
<div data-role = "header">
  <a href = "#" data-role = "button">首页</a>
  <h1>欢迎来到我的主页</h1>
</div>
```

不过，如果在<h1>元素之后放置按钮链接，那么它不会显示在文本右侧。如需向页眉标题的右侧添加按钮，请规定类名为"ui-btn-right"，如下面的代码。

```
<div data-role = "header">
  <h1>欢迎来到我的主页</h1>
  <a href = "#" data-role = "button" class = "ui-btn-right">搜索</a>
</div>
```

提示：页眉可包含一个或两个按钮，然而页脚没有限制。

2. 页脚栏

与页眉相比，页脚更具伸缩性，它们更实用且多变，所以能够包含所需数量的按钮，如下

面的代码。

```
<div data-role="footer">
    <a href="#" data-role="button">转播到新浪微博</a>
    <a href="#" data-role="button">转播到腾讯微博</a>
    <a href="#" data-role="button">转播到 QQ 空间</a>
</div>
```

注释：页脚与页眉的样式不同(它会减去一些内边距和空白，并且按钮不会居中)。如果要修正该问题，请在页脚设置类名为"ui-btn"，如下面的代码。

```
<div data-role="footer" class="ui-btn">
```

可以选择在页脚中水平还是垂直地组合按钮，如下面的代码，显示效果如图 6.3 所示。

```
<div data-role="footer" class="ui-btn">
   <div data-role="controlgroup" data-type="horizontal">
     <a href="#" data-role="button" data-icon="plus">转播到新浪微博</a>
     <a href="#" data-role="button" data-icon="plus">转播到腾讯微博</a>
     <a href="#" data-role="button" data-icon="plus">转播到 QQ 空间</a>
   </div>
</div>
```


图 6.3　页脚中的按钮

3. 定位页眉和页脚

放置页眉和页脚的方式有以下三种。

(1) inline——默认。页眉和页脚与页面内容位于行内。

(2) fixed——页面和页脚会留在页面顶部和底部。

(3) fullscreen——与 fixed 类似，页面和页脚会留在页面顶部和底部。

请使用 data-position 属性来定位页眉和页脚。

(1) inline 定位(默认)，如下面的代码。

```
<div data-role="header" data-position="inline"></div>
<div data-role="footer" data-position="inline"></div>
```

(2) fixed 定位，如下面的代码。

```
<div data-role="header" data-position="fixed"></div>
<div data-role="footer" data-position="fixed"></div>
```

(3) fullscreen 定位：如果需要启用全面定位，请使用 data-position="fixed"，并向该元素添加 data-fullscreen 属性，如下面的代码。

```
<div data-role="header" data-position="fixed" data-fullscreen="true"></div>
<div data-role="footer" data-position="fixed" data-fullscreen="true"></div>
```

提示：fullscreen 对于照片、图像和视频非常理想，对于 fixed 和 fullscreen 定位，触摸屏幕将隐藏和显示页眉及页脚。

6.7 jQuery Mobile 导航栏

导航栏由一组水平排列的链接构成，通常位于页眉或页脚内部。默认地，导航栏中的链接会自动转换为按钮(无须 data-role="button")。请使用 data-role="navbar"属性来定义导航栏，如下面的代码，显示效果如图 6.4 所示。

```
<div data-role="header">
  <div data-role="navbar">
    <ul>
      <li><a href="#anylink">首页</a></li>
      <li><a href="#anylink">页面二</a></li>
      <li><a href="#anylink">搜索</a></li>
    </ul>
  </div>
</div>
```



图 6.4 导航栏

提示：按钮的宽度默认与其内容一致。使用无序列表来均等地划分按钮：一个按钮占据 100%的宽度，两个按钮各分享 50%的宽度，三个按钮 33.3%，以此类推。不过，如果在导航栏中规定了 5 个以上的按钮，那么它会弯折为多行。

活动按钮：当导航栏中的链接被单击时，会获得选取外观(按下)。

如需在不按链接时实现此外观，请使用 class="ui-btn-active"，如下面的代码，显示效果如图 6.5 所示。

```
<li><a href="#anylink" class="ui-btn-active">首页</a></li>
```



图 6.5 活动按钮

对于多个页面，可以为每个按钮设置"被选"外观，以表示用户正在浏览该页面。如果要这么做，请向链接添加 ui-state-persist 类，以及 ui-btn-active 类，如下面的代码。

```
<li><a href="#anylink" class="ui-btn-active ui-state-persist">首页</a></li>
```

6.8　jQuery Mobile 可折叠

可折叠(Collapsibles)允许隐藏或显示内容，对于存储部分信息很有用。如需创建可折叠的内容块，请向某个容器分配 data-role="collapsible"属性。在容器(div)中，添加一个标题元素(h1～h6)，其后是需要扩展的任意 HTML 标记，如下面的代码，显示效果如图 6.6 所示。

```
<div data-role="collapsible">
  <h1>点击我 - 我可以折叠!</h1>
    <p>我是可折叠的内容.</p>
</div>
```

图 6.6　隐藏可折叠内容块

默认地，该内容是关闭的。如需在页面加载时扩展内容，请使用这个属性 data-collapsed="false"，如下面的代码，显示效果如图 6.7 所示。

```
<div data-role="collapsible" data-collapsed="false">
    <h1>点击我 - 我可以折叠!</h1>
    <p>现在我默认是展开的。</p>
</div>
```

图 6.7 显示可折叠内容块

嵌套的可折叠块：可折叠块是可以嵌套的，如下面的代码，显示效果如图 6.8 所示。

```
<div data-role="collapsible">
    <h1>点击我 - 我可以折叠!</h1>
    <p>我是被展开的内容。</p>
    <div data-role="collapsible">
      <h1>点击我 - 我是嵌套的可折叠块!</h1>
      <p>我是嵌套的可折叠块中被展开的内容。</p>
    </div>
</div>
```

提示：可折叠内容块可以被嵌套任意次数。



图 6.8　嵌套的可折叠内容块

可折叠集合：可折叠集合指的是被组合在一起的可折叠块。当新块被打开时，所有其他块会关闭。

首先创建若干内容块，然后通过 data-role="collapsible-set"用新容器包装这个可折叠块，如下面的代码，显示效果如图 6.9 所示。

```
<div data-role="content">
    <div data-role="collapsible-set">
      <div data-role="collapsible">
        <h3>点击我 - 我可以折叠!</h3>
         <p>我是可折叠的内容.</p>
      </div>
      <div data-role="collapsible">
        <h3>点击我 - 我可以折叠!</h3>
        <p>我是可折叠的内容.</p>
      </div>
      <div data-role="collapsible">
        <h3>点击我 - 我可以折叠!</h3>
        <p>我是可折叠的内容.</p>
      </div>
      <div data-role="collapsible">
        <h3>点击我 - 我可以折叠!</h3>
        <p>我是可折叠的内容.</p>
      </div>
    </div>
</div>
```

点击我 - 我可以折叠!
点击我 - 我可以折叠!
点击我 - 我可以折叠!
点击我 - 我可以折叠!

图 6.9　可折叠集合

6.9　jQuery Mobile 网格

jQuery Mobile 提供了一套基于 CSS 的列布局方案。不过，一般不推荐在移动设备上使用列布局，这是由于移动设备的屏幕宽度所限。但是有时需要定位更小的元素，例如按钮或导航栏，就像在表格中那样并排。这时，列布局就恰如其分。网格中的列是等宽的（总宽

是 100%)，无边框、背景、外边距或内边距。可使用的布局网格有 4 种，见表 6.5。

表 6.5　4 种布局网格

网格类	列	列宽度	对应
ui-grid-a	2	50% / 50%	ui-block-a|b
ui-grid-b	3	33% / 33% / 33%	ui-block-a|b|c
ui-grid-c	4	25% / 25% / 25% / 25%	ui-block-a|b|c|d
ui-grid-d	5	20% / 20% / 20% / 20% / 20%	ui-block-a|b|c|d|e

提示：在列容器中，根据不同的列数，子元素可设置类 ui-block-a|b|c|d|e。这些列将依次并排浮动。例如对于 ui-grid-a 类(两列布局)，必须规定两个子元素 ui-block-a 和 ui-block-b；对于 ui-grid-b 类(三列布局)，请添加三个子元素 ui-block-a、ui-block-b 和 ui-block-c。

定制网格：可以通过使用 CSS 来定制列块，如下面的代码，显示效果如图 6.10 所示。

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.2.
min.css">
<script src="http://code.jquery.com/jquery-1.8.3.min.js"></script>
<script src="http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.2.min.js"></script>
<style>
.ui-block-a,
.ui-block-b,
.ui-block-c
{
background-color: lightgray;
border: 1px solid black;
height: 100px;
font-weight: bold;
text-align: center;
padding: 30px;
}
</style>
</head>
<body>

<div data-role="page" id="pageone">
  <div data-role="header">
  <h1>自定义的列</h1>
  </div>

  <div data-role="content">
   <p>三列样式布局:</p>
   <div class="ui-grid-b">
     <div class="ui-block-a"><span>第一个列</span></div>
```

```
      <div class="ui-block-b"><span>第二个列</span></div>
      <div class="ui-block-c"><span>第三个列</span></div>
    </div>
  </div>
</div>

</body>
</html>
```



图 6.10　定制列块

当然，也可以通过使用行内样式来定制块，代码如下。

```
<div class="ui-block-a" style="border: 1px solid black;">
  <span>Text..</span>
</div>
```

多行：在列中包含多个行也是可以的，如下面的代码，显示效果如图 6.11 所示。

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.2.
min.css">
<script src="http://code.jquery.com/jquery-1.8.3.min.js"></script>
<script src="http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.2.min.js"></script>
</head>
<body>

<div data-role="page" id="pageone">
  <div data-role="header">
  <h1>多列</h1>
  </div>

  <div data-role="content">
    <p>三列布局：</p>
    <div class="ui-grid-b">
      <div class="ui-block-a" style="border: 1px solid black;"><span>Some Text</span>
</div>
      <div class="ui-block-b" style="border: 1px solid black;"><span>Some Text</span>
</div>
```

```
      <div class="ui-block-c" style="border: 1px solid black;"><span>Some Text</span>
</div>
    </div>

    <p>多行的三列布局: </p>
    <div class="ui-grid-b">
      <div class="ui-block-a" style="border: 1px solid black;"><span>Some Text</span>
</div>
      <div class="ui-block-b" style="border: 1px solid black;"><span>Some Text</span>
</div>
      <div class="ui-block-c" style="border: 1px solid black;"><span>Some Text</span>
</div>
      <div class="ui-block-a" style="border: 1px solid black;"><span>Some Text</span>
</div>
      <div class="ui-block-b" style="border: 1px solid black;"><span>Some Text</span>
</div>
      <div class="ui-block-a" style="border: 1px solid black;"><span>Some Text</span>
</div>
    </div>
   </div>
</div>

</body>
</html>
```


图 6.11　列中包含多行

6.10　jQuery Mobile 列表

jQuery Mobile 中的列表视图是标准的 HTML 列表：有序列表(<ol>)和无序列表(<ul>)。如需创建列表，请向<ol>或<ul>元素添加 data-role="listview"。如需使这些项目可单击，请在每个列表项(<li>)中规定链接，如下面的代码，显示效果如图 6.12 所示。

```
<div data-role="page" id="pageone">
  <div data-role="content">
    <h2>有序列表: </h2>
    <ol data-role="listview">
```

```
        <li><a href="#">列表项</a></li>
        <li><a href="#">列表项</a></li>
        <li><a href="#">列表项</a></li>
      </ol>
      <h2>无序列表:</h2>
      <ul data-role="listview">
        <li><a href="#">列表项</a></li>
        <li><a href="#">列表项</a></li>
        <li><a href="#">列表项</a></li>
      </ul>
    </div>
  </div>
```

有序列表:
1. 列表项
2. 列表项
3. 列表项

无序列表:
- 列表项
- 列表项
- 列表项

图 6.12 列表

如需为列表添加圆角和外边距,请使用 data-inset="true"属性,如下代码。

```
<ul data-role="listview" data-inset="true">
    <li><a href="#">列表项</a></li>
    <li><a href="#">列表项</a></li>
    <li><a href="#">列表项</a></li>
</ul>
```

提示:默认地,列表中的列表项会自动转换为按钮(无须 data-role="button")。

列表分隔符:列表分隔符(List Dividers)用于把项目组织和组合为分类/节。如需规定列表分隔符,请向<li>元素添加 data-role="list-divider"属性,如下面的代码,显示效果如图 6.13 所示。

```
<div data-role="content">
    <h2>列表分隔符</h2>
    <ul data-role="listview">
      <li data-role="list-divider">欧洲</li>
      <li><a href="#">德国</a></li>
      <li><a href="#">英国</a></li>
      <li data-role="list-divider">亚洲</li>
```

```
        <li><a href="#">中国</a></li>
        <li><a href="#">印度</a></li>
        <li data-role="list-divider">非洲</li>
        <li><a href="#">埃及</a></li>
        <li><a href="#">南非</a></li>
      </ul>
  </div>
```


图 6.13　列表分隔符

如果列表是字母顺序的(例如通讯录),jQuery Mobile 会自动添加恰当的分隔符,在<ol>或<ul>元素上设置 data-autodividers="true"属性,如下面的代码,显示效果如图 6.14 所示。

```
<div data-role="page" id="pageone">
  <div data-role="content">
  <h2>我的通讯录</h2>
  <ul data-role="listview" data-autodividers="true" data-inset="true">
    <li><a href="#">Adele</a></li>
    <li><a href="#">Agnes</a></li>
    <li><a href="#">Albert</a></li>
    <li><a href="#">Billy</a></li>
    <li><a href="#">Bob</a></li>
    <li><a href="#">Calvin</a></li>
    <li><a href="#">Cameron</a></li>
    <li><a href="#">Chloe</a></li>
    <li><a href="#">Christina</a></li>
    <li><a href="#">Diana</a></li>
    <li><a href="#">Gabriel</a></li>
    <li><a href="#">Glen</a></li>
    <li><a href="#">Ralph</a></li>
    <li><a href="#">Valarie</a></li>
  </ul>
   </div>
</div>
```



图 6.14　字母顺序的列表

提示： data-autodividers＝"true"属性通过对列表项文本的首字母进行大写来创建分隔符。

搜索过滤器： 如需在列表中添加搜索框，请使用 data-filter＝"true"属性，如下面的代码，显示效果如图 6.15 所示。

```
<div data-role="page" id="pageone">
  <div data-role="content">
  <h2>我的通讯录</h2>
  <ul data-role="listview" data-autodividers="true" data-inset="true" data-filter=
"true">
    <li><a href="#">Adele</a></li>
    <li><a href="#">Agnes</a></li>
    <li><a href="#">Albert</a></li>
    <li><a href="#">Billy</a></li>
    <li><a href="#">Bob</a></li>
    <li><a href="#">Calvin</a></li>
    <li><a href="#">Cameron</a></li>
    <li><a href="#">Chloe</a></li>
    <li><a href="#">Christina</a></li>
    <li><a href="#">Diana</a></li>
    <li><a href="#">Gabriel</a></li>
    <li><a href="#">Glen</a></li>
```

```
        <li><a href="#">Ralph</a></li>
        <li><a href="#">Valarie</a></li>
      </ul>
    </div>
</div>
```

图 6.15　列表中添加搜索框

默认地，搜索框中的文本是"Filter items..."。如需修改默认文本，请使用 data-filter-placeholder 属性，如以下代码所示。

```
<ul data-role="listview" data-filter="true" data-filter-placeholder="搜索姓名">
```

列表缩略图：为了实现缩略图，请在链接中添加<img>元素，对于大于 16×16px 的图像，jQuery Mobile 将自动把图像调整至 80×80px。如下面的代码，显示效果如图 6.16 所示。

```
<div data-role="content">
    <h2>包含缩略图的列表：</h2>
    <ul data-role="listview" data-inset="true">
      <li>
        <a href="#"><img src="/i/chrome.png"></a>
      </li>
```

```
        <li>
          <a href="#"><img src="/i/firefox.png"></a>
        </li>
      </ul>
    </div>
```



图 6.16　列表缩略图

再看一个例子，使用标准 HTML 来填充带有信息的列表，如下面的代码，显示效果如图 6.17 所示。

```
<div data-role="page" id="pageone">
  <div data-role="content">
    <h2>包含缩略图和文本的列表：</h2>
    <ul data-role="listview" data-inset="true">
      <li>
        <a href="#">
        <img src="/i/chrome.png">
        <h2>Google Chrome</h2>
        <p>Google Chrome is a free, open-source web browser. Released in 2008.</p>
        </a>
      </li>
      <li>
        <a href="#">
        <img src="/i/firefox.png">
        <h2>Mozilla Firefox</h2>
        <p>Firefox is a web browser from Mozilla. Released in 2004.</p>
        </a>
      </li>
    </ul>
  </div>
</div>
```



图 6.17　带有信息的列表

列表图标：如需向列表中添加 16×16px 的图标，请向<img>元素中添加 class="ui-li-icon"属性，如下面的代码，显示效果如图 6.18 所示。

```
<div data-role="page" id="pageone">
  <div data-role="content">
    <h2>带有图标的列表：</h2>
    <ul data-role="listview" data-inset="true">
      <li><a href="#"><img src="/i/us.png" alt="USA" class="ui-li-icon">USA</a>
</li>
      <li><a href="#"><img src="/i/gb.png" alt="Great Britain" class="ui-li-icon">
Great Britain</a></li>
      <li><a href="#"><img src="/i/finland.png" alt="Finland" class="ui-li-icon">
Finland</a></li>
      <li><a href="#"><img src="/i/germany.png" alt="Germany" class="ui-li-icon">
Germany</a></li>
      <li><a href="#"><img src="/i/france.png" alt="France" class="ui-li-icon">
France</a></li>
    </ul>
  </div>
</div>
```



图 6.18　带图标的列表

拆分按钮：如需创建带有垂直分隔栏的拆分列表，请在<li>元素内放置两个链接。

jQuery Mobile 会自动为第二个链接添加蓝色箭头图标的样式，链接中的文本(如有)将在用户划过该图标时显示，如下面的代码，显示效果如图 6.19 所示。

```
<div data-role="page" id="pageone">
  <div data-role="content">
    <h2>拆分按钮</h2>
    <ul data-role="listview" data-inset="true">
      <li>
        <a href="#">
        <img src="/i/chrome.png">
        <h2>Google Chrome</h2>
        <p>Google Chrome is a free, open-source web browser. Released in 2008.</p>
        </a>
```

```
            <a href = "#"> Some Text </a>
          </li>
          <li>
            <a href = "#">
            <img src = "/i/firefox.png">
            <h2 > Mozilla Firefox </h2 >
            <p> Firefox is a web browser from Mozilla. Released in 2004.</p>
            </a>
            <a href = "#"> Some Text </a>
          </li>
        </ul>
      </div>
    </div>
```

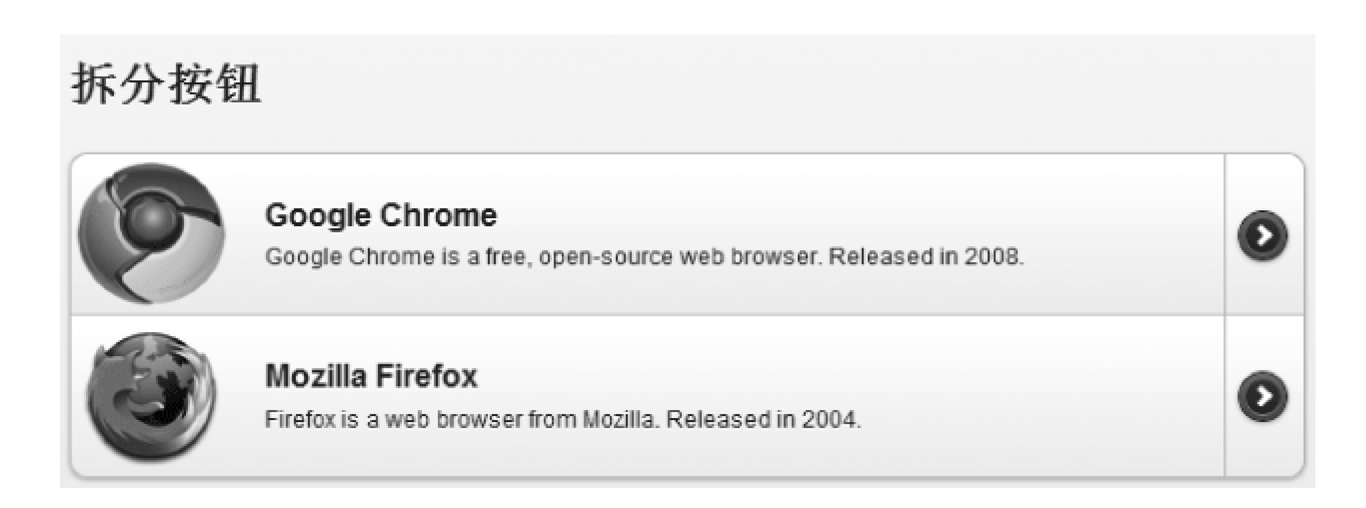


图 6.19　带有垂直分隔栏的拆分列表

通过添加页面和对话框，可使链接的功能更强大，如以下代码所示。

```
<ul data-role = "listview">
  <li>
    <a href = "#"><img src = "chrome.png"></a>
    <a href = "#download" data-rel = "dialog">下载浏览器</a>
  </li>
</ul>
```

计数泡沫：计数泡沫用于显示与列表项相关的数目，例如邮箱中的消息。如需添加计数泡沫，请使用行内元素，比如<span>，设置 class="ui-li-count"属性并添加数字，如下面的代码，显示效果如图 6.20 所示。

```
<div data-role = "page" id = "pageone">
  <div data-role = "content">
    <h2 >我的邮箱</h2 >
    <ul data-role = "listview" data-inset = "true">
      <li><a href = "#">收件箱<span class = "ui-li-count"> 25 </span></a></li>
      <li><a href = "#">发件箱<span class = "ui-li-count"> 432 </span></a></li>
      <li><a href = "#">垃圾箱<span class = "ui-li-count"> 7 </span></a></li>
    </ul>
  </div>
</div>
```

我的邮箱

收件箱 25

发件箱 432

垃圾箱 7

图 6.20　带有计数泡沫列表

6.11　jQuery Mobile 表单

jQuery Mobile 使用 CSS 来设置 HTML 表单元素的样式，以使其更有吸引力、更易用。在 jQuery Mobile 中，可以使用以下表单控件：文本框、搜索框、单选框、复选框、选择菜单、滑动条和翻转切换开关。

当使用 jQuery Mobile 表单时，应该了解以下信息。

(1) <form>元素必须设置 method 和 action 属性。

(2) 每个表单元素必须设置唯一的 id 属性。该 id 在站点的页面中必须是唯一的。这是因为 jQuery Mobile 的单页面导航模型允许许多不同的"页面"同时呈现。

(3) 每个表单元素必须有一个标记(label)。请设置 label 的 for 属性来匹配元素的 id。

例如下面的代码，显示效果如图 6.21 所示。

```
<div data-role="page">
  <div data-role="content">
    <form method="post" action="demoform.asp">
      <label for="fname">姓名: </label>
      <input type="text" name="fname" id="fname">
      <input type="submit" data-inline="true" value="提交">
    </form>
  </div>
</div>
```

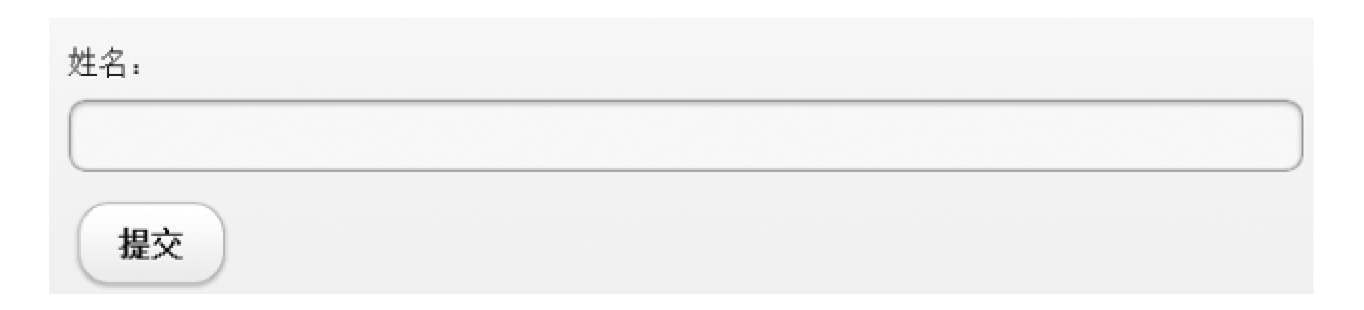


图 6.21　表单

如需隐藏 label，请使用类 ui-hidden-accessible。这很常用，当需要元素的 placeholder 属性充当 label 时，代码如下。

```
<form method="post" action="demoform.asp">
  <label for="fname" class="ui-hidden-accessible">姓名: </label>
  <input type="text" name="fname" id="fname" placeholder="姓名...">
</form>
```

域容器：如果需要 label 和表单元素在宽屏幕上显示正常，请用带有 data-role="fieldcontain"属性的<div>或<fieldset>元素来包装 label 或表单元素，请看如下的代码。

```
<div data-role="page">
  <div data-role="content">
    <form method="post" action="demoform.asp">
      <div data-role="fieldcontain">
        <label for="lname">姓: </label>
        <input type="text" name="lname" id="lname">
        <label for="fname">名: </label>
        <input type="text" name="fname" id="fname">
      </div>
      <input type="submit" data-inline="true" value="提交">
    </form>
  </div>
</div>
```

提示：fieldcontain 属性基于页面宽度来设置 label 和表单控件的样式。当页面宽度大于 480px 时，它会自动将 label 与表单控件放置于同一行。当小于 480px 时，label 会被放置于表单元素之上。如需避免 jQuery Mobile 自动为可单击元素设置样式，请使用 data-role="none"属性，代码如下。

```
<label for="fname">First name:</label>
<input type="text" name="fname" id="fname" data-role="none">
```

文本输入：输入字段是通过标准的 HTML 元素编写的，jQuery Mobile 会为它们设置专门针对移动设备的美观易用的样式。还可以使用新的 HTML5 的<input>类型，如下面的代码，显示效果如图 6.22 所示。

```
<div data-role="content">
    <form method="post" action="demoform.asp">
      <div data-role="fieldcontain">
        <label for="fullname">全名: </label>
        <input type="text" name="fullname" id="fullname">
        <label for="bday">生日: </label>
        <input type="date" name="bday" id="bday">
        <label for="email">电邮: </label>
        <input type="email" name="email" id="email" placeholder="您的邮箱地址..">
      </div>
      <input type="submit" data-inline="true" value="提交">
    </form>
  </div>
```

提示：请使用 placeholder 来规定简短的提示，以描述输入字段的预期值。

图 6.22　文本输入框

文本框：请使用<textarea>来实现多行文本输入。注释：文本框会自动扩大，以适应输入的文本行。如下面的代码，显示效果如图 6.23 所示。

```
<form method="post" action="demoform.asp">
    <div data-role="fieldcontain">
      <label for="info"> Additional Information:</label>
      <textarea name="addinfo" id="info"></textarea>
    </div>
</form>
```

图 6.23　多行文本框

搜索框：输入类型 type="search"是 HTML5 中的新类型，用于定义供输入搜索词的文本字段。如下面的代码，显示效果如图 6.24 所示。

```
<div data-role="page">
  <div data-role="header">
  <h1>搜索框</h1>
  </div>

  <div data-role="content">
    <form method="post" action="demoform.asp">
      <div data-role="fieldcontain">
        <label for="search">搜索: </label>
        <input type="search" name="search" id="search" placeholder="搜索内容...">
      </div>
      <input type="submit" data-inline="true" value="提交">
    </form>
  </div>
</div>
```

单选按钮：当用户只选择有限数量选项中的一个时，会用到单选按钮。如需创建一套单选按钮，请添加 type="radio"的 input 元素以及相应的 label。在<fieldset>元素中包装单选按钮。也可以增加一个<legend>元素来定义<fieldset>的标题。如下面的代码，显



图 6.24　搜索框

示效果如图 6.25 所示。

提示：请用 data-role="controlgroup"属性来组合这些按钮。

```
<div data-role="page">
  <div data-role="header">
   <h1>单选按钮</h1>
  </div>

  <div data-role="content">
    <form method="post" action="demoform.asp">
      <fieldset data-role="controlgroup">
      <legend>请选择您的性别：</legend>
        <label for="male">男性</label>
        <input type="radio" name="gender" id="male" value="male">
        <label for="female">女性</label>
        <input type="radio" name="gender" id="female" value="female">
      </fieldset>
        <input type="submit" data-inline="true" value="提交">
    </form>
  </div>
</div>
```



图 6.25　单选按钮

复选框：当用户选择有限数量选项中的一个或多个选项时，会用到复选框。如下面的代码，显示效果如图 6.26 所示。

```
<div data-role="page">
  <div data-role="header">
  <h1>复选框</h1>
  </div>
```

```
    <div data-role="content">
      <form method="post" action="demoform.asp">
        <fieldset data-role="controlgroup">
          <legend>请选择您喜爱的颜色：</legend>
            <label for="red">红色</label>
            <input type="checkbox" name="favcolor" id="red" value="red">
            <label for="green">绿色</label>
            <input type="checkbox" name="favcolor" id="green" value="green">
            <label for="blue">蓝色</label>
            <input type="checkbox" name="favcolor" id="blue" value="blue">
        </fieldset>
        <input type="submit" data-inline="true" value="提交">
      </form>
    </div>
  </div>
```

图 6.26　复选框

其他例子：如需对单选框或复选框进行水平分组，请使用 data-type="horizontal"属性。如下面的代码，显示效果如图 6.27 所示。

```
  <div data-role="page">
    <div data-role="header">
    <h1>单选按钮和复选框</h1>
    </div>

    <div data-role="content">
      <form method="post" action="demoform.asp">
        <fieldset data-role="controlgroup" data-type="horizontal">
        <legend>请选择您的性别：</legend>
          <label for="male">男性</label>
          <input type="radio" name="gender" id="male" value="male">
          <label for="female">女性</label>
          <input type="radio" name="gender" id="female" value="female">
        </fieldset>

        <fieldset data-role="controlgroup" data-type="horizontal">
          <legend>请选择您喜爱的颜色：</legend>
            <label for="red">红色</label>
```

```
                <input type = "checkbox" name = "favcolor" id = "red" value = "red">
                <label for = "green">绿色</label>
                <input type = "checkbox" name = "favcolor" id = "green" value = "green">
                <label for = "blue">蓝色</label>
                <input type = "checkbox" name = "favcolor" id = "blue" value = "blue">
            </fieldset>
            <input type = "submit" data-inline = "true" value = "提交">
        </form>
    </div>
</div>
```

图 6.27　对单选按钮或复选框进行水平分组

当然，也可以使用域容器来包装<fieldset>，如下面的代码。

```
<div data-role = "fieldcontain">
  <fieldset data-role = "controlgroup">
    <legend>Choose your gender:</legend>
  </fieldset>
</div>
```

如果希望“预选”其中一个按钮，请使用 HTML<input>标签的 checked 属性，如下面的代码。

```
<input type = "radio" checked>
<input type = "checkbox" checked>
```

选择菜单：<select>元素创建带有若干选项的下拉菜单。<select>元素中的<option>元素定义列表中的可用选项。如下面的代码，显示效果如图 6.28 所示。

```
<div data-role = "page">
  <div data-role = "header">
  <h1>选择菜单</h1>
  </div>

  <div data-role = "content">
    <form method = "post" action = "demoform.asp">
      <fieldset data-role = "fieldcontain">
        <label for = "day">选择天</label>
```

```
          < select name = "day" id = "day">
           < option value = "mon">星期一</option >
           < option value = "tue">星期二</option >
           < option value = "wed">星期三</option >
           < option value = "thu">星期四</option >
           < option value = "fri">星期五</option >
           < option value = "sat">星期六</option >
           < option value = "sun">星期日</option >
          </select >
        </fieldset >
        < input type = "submit" data - inline = "true" value = "提交">
      </form >
    </div >
  </div >
```



图 6.28　选择菜单

提示：如果列表中包含一长列相关的选项，请在<select>中使用<optgroup>元素，如下面的代码。

```
< select name = "day" id = "day">
  < optgroup label = "Weekdays">
      < option value = "mon"> Monday </option >
      < option value = "tue"> Tuesday </option >
      < option value = "wed"> Wednesday </option >
  </optgroup >
  < optgroup label = "Weekends">
      < option value = "sat"> Saturday </option >
      < option value = "sun"> Sunday </option >
  </optgroup >
</select >
```

自定义选择菜单：图 6.28 展示了移动平台显示选择菜单的独特方式。如果希望在所有移动设备上显示一致外观的选择菜单，请使用 jQuery 的自定义选择菜单，设置 data-native-menu="false"属性。

```
< select name = "day" id = "day" data - native - menu = "false">
```

Multiple Selections：如需在选择菜单中选取多个选项，请在<select>元素中使用 multiple 属性，选择时的效果如图 6.29 所示。

```
<select name = "day" id = "day" multiple data - native - menu = "false">
```

您可以选择多天：

星期一

星期二

星期三

星期四

星期五

星期六

星期日

图 6.29　自定义选择菜单

滑块控件：滑块允许从一定范围内的数字中选取值。如需创建滑块，请使用<input type="range">。如下面的代码，显示效果如图 6.30 所示。

```
<div data - role = "page">
  <div data - role = "header">
  <h1 >滑块控件</h1 >
  </div >

  <div data - role = "content">
    <form method = "post" action = "demoform.asp">
      <div data - role = "fieldcontain">
        <label for = "points"> Points:</label >
        <input type = "range" name = "points" id = "points" value = "50" min = "0" max = "100">
      </div >
      <input type = "submit" data - inline = "true" value = "提交">
    </form >
  </div >
</div >
```



图 6.30　滑块控件

使用下列属性来规定限定。

max——规定允许的最大值。

min——规定允许的最小值。

step——规定合法的数字间隔。

value——规定默认值。

提示：如果希望突出显示滑块控件的这段轨道，请添加 data-highlight="true"，如下面的代码。

```
<input type="range" data-hightlight="true">
```

切换开关：切换开关常用于开/关或对/错按钮。如需创建切换，请使用 data-role="slider"的<select>元素，并添加两个<option>元素。如下面的代码，显示效果如图 6.31 所示。

```
<div data-role="page">
  <div data-role="content">
    <form method="post" action="demoform.asp">
      <div data-role="fieldcontain">
        <label for="switch">切换开关: </label>
        <select name="switch" id="switch" data-role="slider">
          <option value="on">On</option>
          <option value="off">Off</option>
        </select>
      </div>
      <input type="submit" data-inline="true" value="提交">
    </form>
  </div>
</div>
```



图 6.31　切换开关

提示：请使用 selected 属性来把选项之一设置为预选(突出显示)，如下面的代码。

```
<option value="off" selected>Off</option>
```

6.12　jQuery Mobile 主题

jQuery Mobile 提供了 5 种不同的样式主题，从 a 到 e，如表 6.6 所示。每种主题带有不同颜色的按钮、栏、内容块，等等。jQuery Mobile 中的一种主题由多种可见的效果和颜色构成。

如需定制应用程序的外观，请使用 data-theme 属性，并为该属性分配一个字母，如下面的代码。

```
<div data-role="page" data-theme="a|b|c|d|e">
```

表 6.6　样式主题

值	描　　述
a	默认。黑色背景上的白色文本
b	蓝色背景上的白色文本/灰色背景上的黑色文本
c	亮灰色背景上的黑色文本
d	白色背景上的黑色文本
e	橙色背景上的黑色文本

默认地，jQuery Mobile 为页眉和页脚使用 a 主题，为页眉内容使用 c 主题(亮灰)。不过也可以对主题进行混合。

主题化的页面、内容和页脚：如下面的代码，显示效果如图 6.32 所示。

```
<div data-role="page" id="pageone">
  <div data-role="header" data-theme="b">
    <h1>此处是页面标题</h1>
  </div>

  <div data-role="content" data-theme="a">
    <p>此处是内容.</p>
    <a href="#" data-role="button">按钮</a>
    <p>此处是 <a href="#">链接</a>!</p>
  </div>

  <div data-role="footer" data-theme="e">
    <h1>此处是页脚文本</h1>
  </div>
</div>
```



图 6.32　主题化的页面、内容和页脚

主题化的对话框：如下面的代码，显示效果如图 6.33 所示。

```
<div data-role="page" id="pageone">
  <div data-role="header">
    <h1>欢迎来到我的主页</h1>
  </div>

  <div data-role="content">
    <p>这是一张标准页面。</p>
    <a href="#pagetwo" data-rel="dialog">转到主题化的对话页面</a>
```

```
  </div>

  <div data-role="footer">
    <h1>页脚文本</h1>
  </div>
</div>

<div data-role="page" id="pagetwo" data-overlay-theme="e">
  <div data-role="header" data-theme="b">
    <h1>我是主题化的对话框</h1>
  </div>

  <div data-role="content" data-theme="a">
    <p>data-overlay-theme 属性规定对话框出现在其上的页面的背景色。</p>
    <a href="#pageone">转到页面一</a>
  </div>

  <div data-role="footer" data-theme="c">
    <h1>页脚文本</h1>
  </div>
</div>
```



图 6.33　主题化的对话框

主题化的按钮：如下面的代码，显示效果如图 6.34 所示。

```
<div data-role="page" id="pageone">
  <div data-role="header">
    <h1>此处是页面标题</h1>
  </div>

  <div data-role="content" data-theme="e">
    <p>此处是页面内容。</p>
    <a href="#" data-role="button">按钮</a>
  <p>默认地，页面的子元素会继承应用主题的父元素的样式 - 在这种情况下，上面的按钮会被自
动赋予的 data-theme 为 "e"。</p>

  <p>如需人工添加按钮的样式，请在链接中添加 data-theme 属性：</p>
    <a href="#" data-role="button" data-theme="a">按钮</a>
    <a href="#" data-role="button" data-theme="b">按钮</a>
    <a href="#" data-role="button" data-theme="c">按钮</a>
  </div>

  <div data-role="footer">
    <h1>页脚文本</h1>
```

```
    </div>
</div>
```



图 6.34　主题化的按钮

主题化的图标：如下面的代码，显示效果如图 6.35 所示。

```
<div data-role="page" id="pageone">
    <div data-role="content">
        <a href="#" data-role="button" data-icon="arrow-l" data-iconpos="notext" data-theme="a">Left Arrow Icon</a>
        <a href="#" data-role="button" data-icon="arrow-r" data-iconpos="notext" data-theme="b">Right Arrow Icon</a>
        <a href="#" data-role="button" data-icon="arrow-u" data-iconpos="notext" data-theme="c">Up Arrow Icon</a>
        <a href="#" data-role="button" data-icon="arrow-d" data-iconpos="notext" data-theme="d">Down Arrow Icon</a>
        <a href="#" data-role="button" data-icon="plus" data-iconpos="notext" data-theme="e">Plus Icon</a>
    </div>
</div>
```



图 6.35　主题化的图标

页眉和页脚中的主题化按钮：如下面的代码，显示效果如图 6.36 所示。

```
<div data-role="page" id="pageone">
  <div data-role="header">
    <a href="#" data-role="button" data-icon="home" data-theme="b">首页</a>
    <h1>欢迎来到我的主页</h1>
    <a href="#" data-role="button" data-icon="search" data-theme="e">搜索</a>
  </div>

  <div data-role="content">
   <p>此处是内容...</p>
  </div>

  <div data-role="footer">
  <a href="#" data-role="button" data-theme="b" data-icon="plus">转播到新浪微博
</a>
  <a href="#" data-role="button" data-theme="c" data-icon="plus">转播到腾讯微博
</a>
  <a href="#" data-role="button" data-theme="e" data-icon="plus">转播到 QQ 空间
</a>
  </div>
</div>
```



图 6.36　页眉和页脚中的主题化按钮

主题化的导航栏：如下面的代码，显示效果如图 6.37 所示。

```
<div data-role="page" id="pageone">
  <div data-role="header">
    <h1>此处是页面标题</h1>
  </div>

  <div data-role="content">
    <p>工具栏中的导航栏将自动继承其父元素的样式。如需定制每个按钮的外观，请在链接内添
加 data-theme 属性。</p>
  </div>

  <div data-role="footer" data-theme="e">
  <h1>此处是页脚文本</h1>
    <div data-role="navbar">
      <ul>
        <li><a href="#" data-icon="home" data-theme="b">按钮 1</a></li>
        <li><a href="#" data-icon="arrow-r">按钮 2</a></li>
```

```
        <li><a href="#" data-icon="arrow-r">按钮 3</a></li>
        <li><a href="#" data-icon="search" data-theme="a">按钮 4</a></li>
      </ul>
    </div>
  </div>
</div>
```



图 6.37　主题化的导航栏

主题化的可折叠按钮和内容：如下面的代码，显示效果如图 6.38 所示。

```
<div data-role="content">
    <div data-role="collapsible" data-theme="b" data-content-theme="e">
      <h1>点击我 - 我是可折叠的!</h1>
      <p>我是可折叠内容。</p></div>
</div>
```

图 6.38　主题化的可折叠按钮和内容

主题化列表：如下面的代码，显示效果如图 6.39 所示。

```
<div data-role="page" id="pageone">
  <div data-role="content">
    <h2>有序列表</h2>
    <ol data-role="listview" data-theme="b">
      <li><a href="#">列表项</a></li>
      <li><a href="#">列表项</a></li>
      <li data-theme="a"><a href="#">列表项</a></li>
      <li><a href="#">列表项</a></li>
    </ol>
    <br>
    <h2>无序列表</h2>
    <ul data-role="listview" data-theme="e">
      <li><a href="#">列表项</a></li>
      <li data-theme="a"><a href="#">列表项</a></li>
      <li data-theme="b"><a href="#">列表项</a></li>
      <li><a href="#">列表项</a></li>
```

```
    </ul>
    <br>
  </div>
</div>
```

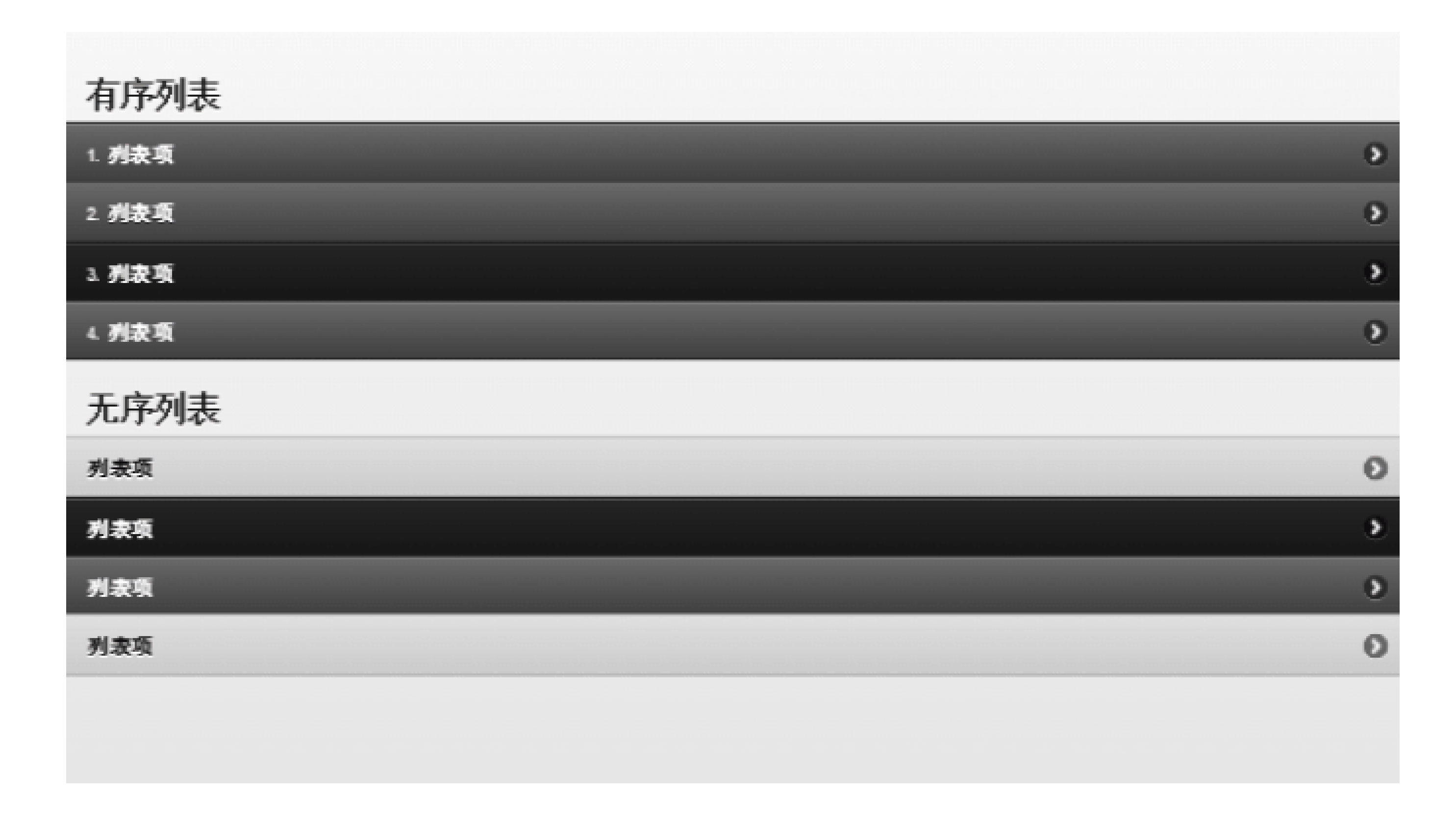


图 6.39　主题化的列表

主题化划分按钮：如下面的代码，显示效果如图 6.40 所示。

```
<div data-role="page" id="pageone">
  <div data-role="content">
    <h2>划分按钮实例</h2>
    <ul data-role="listview" data-inset="true" data-split-theme="e">
      <li data-role="divider" data-theme="a">浏览器</li>
      <li>
        <a href="#">
        <img src="/i/chrome.png">
        <h2>Google Chrome</h2>
        <p>Google Chrome 是一款免费的开源浏览器。发布于 2008 年。</p>
        </a>
        <a href="#download" data-rel="dialog" data-transition="pop">下载浏览器</a>
      </li>
      <li>
        <a href="#">
        <img src="/i/firefox.png">
        <h2>Mozilla Firefox</h2>
        <p>Firefox 是一款来自 Mozilla。发布于 2004 年。</p>
        </a>
        <a href="#download" data-rel="dialog" data-transition="pop">下载浏览器</a>
      </li>
    </ul>
  </div>
</div>

<div data-role="page" id="download" data-overlay-theme="e">
  <div data-role="content">
```

```
    <h3>划分按钮实例</h3>
      <p>下面的按钮仅供演示。</p>
      <a href="#" data-role="button" data-rel="back" data-theme="b" data-icon=
"check" data-inline="true" data-mini="true">下载</a>
      <a href="#" data-role="button" data-rel="back" data-inline="true" data-mini
="true">取消</a>
    </div>
  </div>
```



图 6.40　主题化划分按钮

主题化的可折叠列表：如下面的代码，显示效果如图 6.41 所示。

```
<div data-role="page" id="pageone">
  <div data-role="header">
  <h1>主题化的可折叠列表</h1>
  </div>

  <div data-role="content">
    <div data-role="collapsible" data-theme="b" data-content-theme="e">
    <h4>A</h4>
    <ul data-role="listview">
      <li><a href="#">Adam</a></li>
      <li><a href="#">Angela</a></li>
    </ul>
  </div>

    <div data-role="collapsible" data-theme="b" data-content-theme="a">
    <h4>B</h4>
    <ul data-role="listview">
       <li><a href="#">Bill</a></li>
       <li><a href="#">Bob</a></li>
    </ul>
    </div>

    <div data-role="collapsible" data-theme="b" data-content-theme="d">
    <h4>C</h4>
    <ul data-role="listview">
      <li><a href="#">Calvin</a></li>
      <li><a href="#">Cameron</a></li>
```

```
      <li><a href="#">Christina</a></li>
    </ul>
    </div>
  </div>

  <div data-role="footer">
  <h1>此处是页脚文本</h1>
  </div>
</div>
```



图 6.41　主题化的可折叠列表

主题化表单：如下面的代码，显示效果如图 6.42 所示。

```
<div data-role="page" id="pageone">
  <div data-role="header">
  <h1>主题化表单</h1>
  </div>

  <div data-role="content" data-theme="e">
  <form method="post" action="demoform.asp">
  <div data-role="fieldcontain">
    <label for="name">全名: </label>
    <input type="text" name="text" id="name" placeholder="您的姓名..." data-theme="a">
    <br><br>

    <label for="search">您需要搜索什么?</label>
    <input type="search" name="search" id="search" placeholder="需要搜索的内容..." data-
theme="d">
    <br><br>

    <label for="date">今天的日期: </label>
    <input type="date" name="date" id="date">
    <br><br>

    <label for="colors">请选择喜爱的颜色: </label>
    <select id="colors" name="colors" data-theme="b">
      <option value="red">红色</option>
      <option value="green">绿色</option>
      <option value="blue">蓝色</option>
```

```
        </select>
        <br><br>

        <label for="switch">切换开关: </label>
        <select name="switch" id="switch" data-role="slider" data-theme="a">
          <option value="on">On</option>
          <option value="off">Off</option>
        </select>
        <br><br>

      <div data-role="controlgroup">
      <legend>请选择喜爱的电影: </legend>
      <label for="mov1">蜘蛛侠</label>
      <input type="checkbox" name="mov1" id="mov1" data-theme="a">
      <label for="mov2">变形金刚</label>
      <input type="checkbox" name="mov2" id="mov2" data-theme="b">
      <label for="mov3">星球大战</label>
      <input type="checkbox" name="mov3" id="mov3" data-theme="c">
      </div>
    </div>
    <input type="submit" data-inline="true" value="提交">
    </form>
    </div>
</div>
```

图 6.42　主题化的表单

主题化的可折叠表单：如下面的代码，显示效果如图 6.43 所示。

```
<div data-role="content">
    <form method="post" action="demoform.asp">
```

```
          <fieldset data-role="collapsible" data-theme="b" data-content-theme="e">
            <legend>点击我 - 我是可折叠的!</legend>
              <label for="name">全名: </label>
              <input type="text" name="text" id="name">
              <p>喜爱的颜色: </p>
            <div data-role="controlgroup">
              <label for="red">红色</label>
              <input type="checkbox" name="favcolor" id="red" value="red">
              <label for="green">绿色</label>
              <input type="checkbox" name="favcolor" id="green" value="green">
              <label for="blue">蓝色</label>
              <input type="checkbox" name="favcolor" id="blue" value="blue">
            </div>
          <input type="submit" data-inline="true" value="提交" data-theme="a">
          </fieldset>
        </form>
    </div>
```

图 6.43　主题化的可折叠表单

添加新主题：jQuery Mobile 同时允许向移动页面添加新主题。

请通过编辑 CSS 文件(如已下载 jQuery Mobile)来添加或编辑新主题。只需复制一段样式,并用字母名(f-z)来对类进行重命名,然后调整为用户喜欢的颜色和字体即可。

还可以通过在 HTML 文档中使用主题类来添加新样式,为工具条添加类 ui-bar-(a-z),并为内容添加类 ui-body-(a-z),如以下代码所示。

```
<style>
.ui-bar-f
{
color:green;
background-color:yellow;
}
.ui-body-f
{
font-weight:bold;
color:purple;
```

```
}
</style>

<div data-role="page">
  <div data-role="header" data-theme="f">
    <h1>应用 "f" 主题的标题</h1>
  </div>

  <div data-role="content" data-theme="f">
    <p>应用新的 "f" 主题的内容!</p>
  </div>
</div>
```

6.13 jQuery Mobile 事件

jQuery Mobile 还提供若干种为移动浏览定制的事件。

(1) 触摸事件——当用户触摸屏幕时触发(敲击和滑动)。

(2) 滚动事件——当上下滚动时触发。

(3) 方向事件——当设备垂直或水平旋转时触发。

(4) 页面事件——当页面被显示、隐藏、创建、加载以及/或卸载时触发。

初始化 jQuery Mobile 事件:在 jQuery 中,可以使用文档 ready 事件来阻止 jQuery 代码在文档结束加载(is ready)前运行,如下面的代码。

```
<script>
$(document).ready(function(){
  $("p").on("click",function(){
    $(this).hide();
  });
});
</script>

<p>如果您点击我,我会消失。</p>
<p>点击我,我会消失。</p>
<p>点击我,我也会消失。</p>
```

然而,在 jQuery Mobile 中使用 pageinit 事件,该事件在页面已初始化并完善样式设置后发生。如下代码,第二个参数指向("#pageone")指定了事件页面的 id。

```
<script>
$(document).on("pageinit","#pageone",function(){
  $("p").on("click",function(){
    $(this).hide();
  });
});
</script>
...
```

```
<div data-role="page" id="pageone">
  <div data-role="header">
    <h1>页眉文本</h1>
  </div>

  <div data-role="content">
    <p>如果您点击我,我会消失。</p>
    <p>点击我,我会消失。</p>
    <p>点击我,我也会消失。</p>
  </div>

  <div data-role="footer">
    <h1>页脚文本</h1>
  </div>
</div>
```

注释:jQuery on()方法用于添加事件处理程序。

jQuery Mobile Tap 事件:tap 事件在用户敲击某个元素时触发。下面的例子为当<p>元素上触发 tap 事件时,隐藏当前<p>元素。

```
$("p").on("tap",function(){
 $(this).hide();
});
```

jQuery Mobile Taphold 事件:taphold 事件在用户敲击某个元素并保持一秒时被触发,如下面的代码。

```
$("p").on("taphold",function(){
 $(this).hide();
});
```

jQuery Mobile Swipe 事件:swipe 事件在用户在某个元素上水平滑动超过 30px 时被触发,如下面的代码。

```
$("p").on("swipe",function(){
 $("span").text("Swipe detected!");
});
```

jQuery Mobile Swipeleft 事件:swipeleft 事件在用户在某个元素上从左滑动超过 30px 时被触发,如下面的代码。

```
$("p").on("swipeleft",function(){
  alert("You swiped left!");
});
```

jQuery Mobile Swiperight 事件:swiperight 事件在用户在某个元素上从右滑动超过 30px 时被触发,如下面的代码。

```
$ ("p").on("swiperight",function(){
  alert("You swiped right!");
});
```

jQuery Mobile Scrollstart 事件：scrollstart 事件在用户开始滚动页面时被触发，如下面的代码。

```
$ (document).on("scrollstart",function(){
  alert("开始滚动!");
});
```

jQuery Mobile Scrollstop 事件：scrollstop 事件在用户停止滚动页面时被触发，如下面的代码。

```
$ (document).on("scrollstop",function(){
    alert("结束滚动!");
});
```

jQuery Mobile orientationchange 事件：orientationchange 事件在用户垂直或水平旋转移动设备时被触发。如需使用 orientationchange 事件，请把它添加到 window 对象，如下面的代码。

```
$ (window).on("orientationchange",function(){
    alert("方向已改变!");
});
```

callback 函数可以设置一个参数，即 event 对象，它会返回移动设备的方向：portrait（设备被握持的方向是垂直的）或 landscape（设备被握持的方向是水平的），如下面的代码。

```
$ (window).on("orientationchange",function(event){
    alert("方向是: " + event.orientation);
});
```

由于 orientationchange 事件与 window 对象绑定，可以使用 window. orientation 属性来设置不同样式以区分 portrait 和 landscape 视图，如下面的代码。

```
$ (window).on("orientationchange",function(){
    if(window.orientation == 0)       //Portrait
    {
      $ ("p").css({"background-color":"yellow","font-size":"300%"});
     }
     else                             //Landscape
    {
      $ ("p").css({"background-color":"pink","font-size":"200%"});
     }
});
```

提示：window.orientation 属性对 portrait 视图返回 0，对 landscape 视图返回 90 或 −90。

jQuery Mobile 页面事件：在 jQuery Mobile 中与页面打交道的事件被分为以下 4 类。

(1) Page Initialization——在页面创建前，当页面创建时，以及在页面初始化之后触发。

(2) Page Load/Unload——当外部页面加载时、卸载时或遭遇失败时触发。

(3) Page Transition——在页面过渡之前和之后触发。

(4) Page Change——当页面被更改，或遭遇失败时触发。

jQuery Mobile Initialization 事件：当 jQuery Mobile 中的一个典型页面进行初始化时，它会经历三个阶段：在页面创建前、页面创建、页面初始化。每个阶段触发的事件都可用于插入或操作代码，见表 6.7。

表 6.7　jQuery Mobile Initialization 事件

事　件	描　述
pagebeforecreate	当页面即将初始化，并且在 jQuery Mobile 已开始增强页面之前，触发该事件
pagecreate	当页面已创建，但增强完成之前，触发该事件
pageinit	当页面已初始化，并且在 jQuery Mobile 已完成页面增强之后，触发该事件

下面的例子演示在 jQuery Mobile 中创建页面时，何时触发每种事件，如下面的代码。

```
$(document).on("pagebeforecreate",function(event){
  alert("触发 pagebeforecreate 事件!");
});
$(document).on("pagecreate",function(event){
  alert("触发 pagecreate 事件!");
});
$(document).on("pageinit",function(event){
  alert("触发 pageinit 事件!")
});
```

jQuery Mobile Load 事件：页面加载事件属于外部页面。

无论外部页面何时载入 DOM，将触发两个事件。第一个是 pagebeforeload，第二个是 pageload(成功)或 pageloadfailed(失败)。表 6.8 解释了这些事件。

表 6.8　jQuery Mobile Load 事件

事　件	描　述
pagebeforeload	在任何页面加载请求做出之前触发
pageload	在页面已成功加载并插入 DOM 后触发
pageloadfailed	如果页面加载请求失败，则触发该事件。默认地，将显示"Error Loading Page"消息

下面的例子演示了 pageload 和 pageloadfailed 事件的工作原理。

```
$(document).on("pageload",function(event,data){
  alert("触发 pageload 事件!\nURL: " + data.url);
});
```

```
$(document).on("pageloadfailed",function(event,data){
  alert("抱歉,被请求页面不存在。");
});
```

jQuery Mobile 过渡事件：还可以在从一页过渡到下一页时使用事件。页面过渡涉及两个页面：一张“来”的页面和一张“去”的页面，这些过渡使当前活动页面(“来的”页面)到新页面(“去的”页面)的改变过程变得更加动感。表 6.9 解释了这些事件。

表 6.9　jQuery Mobile 过渡事件

事　　件	描　　述
pagebeforeshow	在“去的”页面触发，在过渡动画开始前
pageshow	在“去的”页面触发，在过渡动画完成后
pagebeforehide	在“来的”页面触发，在过渡动画开始前
pagehide	在“来的”页面触发，在过渡动画完成后

下面的代码演示了过渡事件的工作原理。

```
$(document).on("pagebeforeshow","#pagetwo",function(){     //当进入页面二时
  alert("页面二即将显示");
});
$(document).on("pageshow","#pagetwo",function(){           //当进入页面二时
  alert("现在显示页面二");
});
$(document).on("pagebeforehide","#pagetwo",function(){     //当离开页面二时
  alert("页面二即将隐藏");
});
$(document).on("pagehide","#pagetwo",function(){           //当离开页面二时
  alert("现在隐藏页面二");
});
```

小　　结

本章主要介绍了 jQuery Mobile 框架的用户接口和特性，如页面的切换效果、基本页面按钮、图标、工具栏、导航栏、可折叠效果、网格、列表、表单、主题以及事件，以便于开发人员在移动应用上使用。通过使用该框架可以开发出真正的移动 Web 网站。

第 7 章 Sencha Touch

本章学习目标

- Sencha Touch 开发环境搭建
- Sencha Touch 的基本使用

Sencha Touch 是一个移动 HTML5 开发框架(以下简称 ST),可以让 Web App 看起来像 Native App。美丽的用户界面组件和丰富的数据管理,全部基于最新的 HTML5 和 CSS3 的 Web 标准,全面兼容 Android 和 Apple iOS 设备。下面是 ST 官方给出的几点特性。

(1) 基于最新的 Web 标准——HTML5、CSS3、JavaScript。整个库在压缩和 gzip 后大约 80KB,通过禁用一些组件还会使它更小。

(2) 支持世界上最好的设备。兼容 Android 和 iOS,Android 上的开发人员还可以使用一些专为 Android 定制的主题。

(3) 增强的触摸事件。在 touchstart、touchend 等标准事件基础上,增加了一组自定义事件数据集成,如 tap、swipe、pinch、rotate 等。

(4) 数据集成。提供了强大的数据包,通过 AJAX、JSON、YQL 等方式绑定到组件模板,写入本地离线存储。

(5) 采用 ExtJS4 强大的类系统,使开发者可以通过 JavaScript 创建和继承已有的类,类系统提供了继承、依赖加载、强大的配置选项等内容。

7.1 开发环境准备

开发环境不需要准备太多,只需要满足以下的条件。

(1) Sencha touch2 SDK。

(2) 一个支持 HTML5 的浏览器,推荐使用谷歌的 Chrome 和苹果的 Safari。

7.2 Sencha Touch2 SDK

先下载 Sencha Touch 框架(http://www.sencha.com/products/touch/)。

将下载的 Sencha Touch 开发包解压可以看到以下文件,如图 7.1 所示。

builds 目录是打包过后的一些 JS 文件。

command 是 ST 自带的一些打包工具,例如,将 ST 的原生态应用打包成支持 Android

builds		文件夹	2012-3-29 9:42
command		文件夹	2012-3-29 9:41
docs		文件夹	2012-3-29 9:43
examples		文件夹	2012-3-29 9:42
microloader		文件夹	2012-3-29 9:41
pkgs		文件夹	2012-3-29 9:44
resources		文件夹	2012-3-29 9:41
src		文件夹	2012-3-29 9:41
.senchasdk	1 KB	SENCHASDK 文件	2012-3-6 8:39
getting-started.html	1 KB	Chrome HTML Doc...	2012-3-6 8:39
index.html	1 KB	Chrome HTML Doc...	2012-3-6 8:39
license.txt	35 KB	文本文档	2012-3-6 8:41
release-notes.html	127 KB	Chrome HTML Doc...	2012-3-6 8:39
sencha-touch.js	91 KB	JScript Script ...	2012-3-6 8:39
sencha-touch-all.js	568 KB	JScript Script ...	2012-3-6 8:39
sencha-touch-all-debug.js	2,389 KB	JScript Script ...	2012-3-6 8:39
sencha-touch-debug.js	489 KB	JScript Script ...	2012-3-6 8:39
touch.jsb3	57 KB	JSB3 文件	2012-3-6 8:39
version.txt	1 KB	文本文档	2012-3-6 8:39

图 7.1 Sencha Touch 开发包

设备的 apk 安装文件等。

docs 是官方的 API 参考文档，需要部署在 Web 容器中才可以查看。例如 Tomcat。

examples 是 ST 的官方实例，里面包含完整的源代码，基本包括 ST 的所有组件及应用程序的开发样例，因此是学习 ST 的最佳文档。

mircoloader 是 ST2 一个重大的改进，类加载器，优化了 ST 组件的载入速度。

resources 是 ST 的一些资源文件，包括 CSS 样式、主题文件等，例如 ST 给开发者提供了 Android、Apple 等常用移动设备平台的主题，开发中通常也需要引用此目录下的一些样式文件。

src 是 ST 组件的源代码，如图 7.2 所示。

以上几个是 ST 的核心库文件。sencha-touch.js 是包含部分组件的一个库文件，它是经过压缩处理后的库文件；sencha-touch-all.js 则是包含 ST 所有组件的库文件，通常在开发过程中也是引用此库文件；sencha-touch-all-debug.js 顾名思义是用来调试的，因为包含缩进，所以方便调试；sencha-touch-debug.js 也是用来调试的。

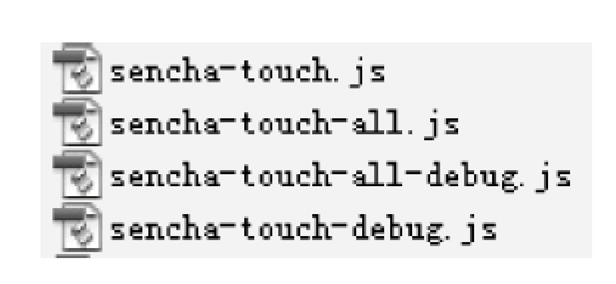


图 7.2 ST 组件的源代码

7.3 框架的加载

至此已经了解了开发 ST 应用所需的环境，也了解了 ST 的官方 SDK 的一些情况，接下来就是如何在自己的项目中引用 ST 框架，开始 ST 开发之旅。

首先在 HTML 页面的 head 标签中间引入 ST 框架的 CSS 样式文件，JS 库文件，如图 7.3 所示。

注意：这里引入的 CSS 及 JS 文件路径应根据自己的目录结构做相应的调整。

app.js 是用户自己定义的 JavaScript 文件，一般一个应用都有一个 app.js 文件作为应



图 7.3　ST 框架

用的启动文件，在这里面会写一些关于应用启动加载的代码，具体的 app.js 如何去写将在稍后章节中详细阐述。接下来看看整个 ST 应用的一个目录结构是怎样的，如图 7.4 所示。

app		文件夹	2012-6-15 13:32
css		文件夹	2012-6-15 13:32
images		文件夹	2012-6-15 13:32
js		文件夹	2012-6-15 13:32
lib		文件夹	2012-6-15 13:32
app.js	1 KB	JScript Script ...	2012-4-16 10:10
index.html	2 KB	Chrome HTML Doc...	2012-7-3 11:19

图 7.4　ST 应用的目录结构

app 目录包含 ST 应用的控制器、模型、代理、数据存储器、视图、设备配置文件等，打开 app 目录如图 7.5 所示。

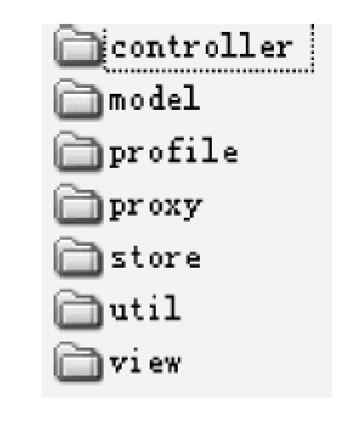


图 7.5　app 目录

controller 目录对应 ST 应用的控制器，按照官方的 MVC 开发模式，所有的控制处理逻辑代码都应该在此目录之下，它类似 J2EE 开发中 MVC 模式的 C，是集中控制应用程序的行为。model 包含一些模型定义文件，对应于 MVC 开发模式的 M，它是应用程序需要用到的模型的定义，它的含义和 J2EE 开发中 MVC 模式的模型意义相似。profile 是配置各个平台的一个启动配置文件，如果应用程序需要在不同的平台及不同的设备上运行，则可能需要在此配置每个设备平台的专属配置文件，例如在 Android、iOS 或者 iPad 等平台及设备上。proxy 是和 ST 应用中数据交互密不可分的，它相当于一个数据获取的来源，充当代理获取数据的一个角色，store 目录则是 ST 的一个核心部分，它是和所有数据存储有关的。view 则是经典 MVC 模式中的 V，它提供了应用程序中视图的部分，用来展现 ST 应用的 UI 部分。这样，model、view、controller 就组成了 ST 应用的一个 MVC 模式，在 ST2 的官方文档及实例中都极力推荐采用 MVC 模式来开发自己的应用，因为这样有一个好处，应用程序的数据、视图、行为控制分离，代码结构清晰，便于日后的维护，同样更适用于实现组件化开发。

7.4　Sencha Touch 应用开发模式之 MVC

MVC 结构是常用的数据结构，应用起来也比较标准。

M 数据模型：在应用程序中表示一种数据模型，比如一个电子商务应用程序可能会有用户、产品、订单等不同的数据模型。

V 视图：负责将数据展示给用户，并扩充 Sencha Touch 的内置组件。（可以理解为用户界面的一个个组成部分。）

C 控制器：处理应用程序的交互，侦听用户的轻触、猛击等事件并做出相应的响应。

S 数据存储器：负责把数据加载到应用并以列表(List)或者数据视图(DataView)等形式表现出来。

P 设备配置：可以为平板电脑和手机等不同设备，轻易定制应用程序用户界面，并尽可能多地共享代码。

Sencha touch2 的 MVC 结构如图 7.6 所示。

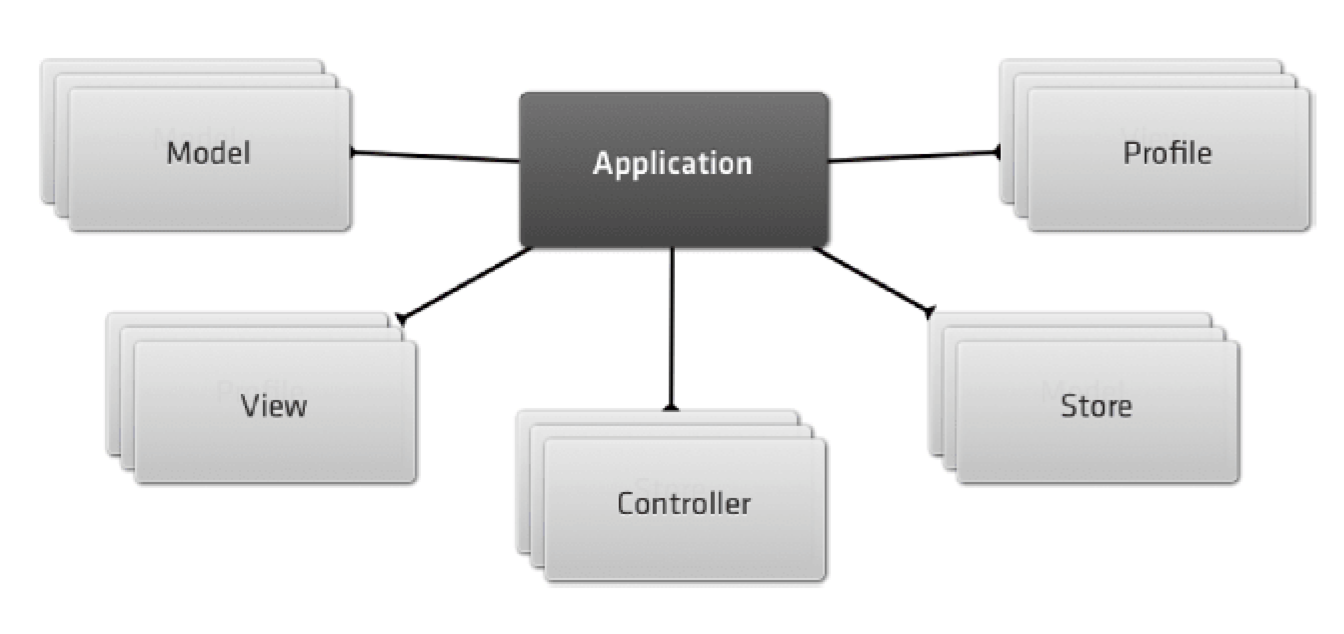


图 7.6　Sencha touch2 的 MVC 结构

对象通常是开发一个 ST 应用时需要定义的第一个东西，类似下面这样。

```
Ext.application({
    name: 'MyApp',
    models: ['User', 'Product', 'nested.Order'],
    views: ['OrderList', 'OrderDetail', 'Main'],
    controllers: ['Orders'],
    launch: function() {
        Ext.create('MyApp.view.Main');
    }
});
```

如代码所示，application 构造参数中有一个 name 属性，它为应用程序创建一个唯一命名空间，其下包含该应用全部的 model、view、controller 还有其他 class(类)，比如一个叫做 MyApp 的应用就应该遵循以下形式来组织：MyApp.model.User、MyApp.controller.Users、MyApp.view.Main 等，这可以保证整个应用程序都处在一个唯一全局变量下，从而最大限度降低代码冲突的可能性。

应用程序会按照在 application 构造函数中定义好的 models，views 和 controllers 属性成员来自动加载它们对应的 class(类)到当前应用，ST2 架构约定的文件结构如下：model 都放在 app/model 目录，controller 都放在 app/controller 目录，view 则放在 app/view 目录，比如 app/model/User.js，app/controller/Orders.js，app/view/Main.js，这样应用程序就可以自动找到并加载这些类，通过这里就基本可以了解到 ST 应用其实就是一个 HTML，一个 app.js 再加上分别处于 app/model、app/view、app/controller 之下的一堆 model、view、controller 文件。除了遵循上述常规命名格式以外，还可以使用完整类名的方式来定义这些配置，换言之，application 构造函数中的 models、views、controllers 这些参数属性都可以用完整类名方式来定义。

7.4.1 控制器

controller 就像胶水一样黏合出一个完整的应用程序，它们侦听 UI 界面触发的事件，然后做出相应的动作，还能够让开发者的代码更简洁，可读性好，从而把界面逻辑和控制逻辑隔离开来。

假如用户通过一个 login 表单来登录应用程序，此时的 view 就是这个包含所有字段和其他元素的表单，而它的 controller 要做的就是侦听表单提交按钮的点触事件并进行身份验证，每次要处理数据或者状态的时候，这个 controller 才是应该起作用的类，而不是 view。

controller 通过一些简单的约定，展示了一套虽小但很强大的特性。应用程序的每一个 controller 都是 Ext. app. Controller 的一个子类，当然用户也可以继承现有的 controller，只要它继承自 Ext. app. Controller，它都存在于 MyApp. controller. * 的命名空间，例如，程序中有一个叫做 Sessions 的 controller，那么它的命名空间就是 MyApp. controller. Sessions 并且被定义在 app/controller/Sessions. js 文件中。

每个 controller 都是 Ext. app. Controller 的一个子类，加载它的应用程序也只会对它实例化一次，应用程序自己会控制每个 controller 在同一时间只有一个实例。用户使用 Application 对象的 controllers 参数来加载 controller 并且会自动实例化它们。

这里将演示如何快速定义上面描述的 Sessions 控制器，将使用 controller 的两个配置参数，refs 和 control。refs 是找到页面组件的简单方式，这个例子里 controller 会搜索所有 formpanel 类型的控件，然后将找到的第一个赋值给 loginForm 属性，这个属性会在下面的 doLogin 方法中用到。

然后要做的是配置 control 参数，像 refs 一样使用 ComponentQuery 选择器来查找所有包含 button 控件的 formpanel 下的 button 控件，在本例中，将会得到登录表单当中的提交按钮，任意一个符合此条件的 button 触发了 tap 事件，controller 都会去调用其中的 doLogin 函数。例子代码如下。

```
Ext.define('MyApp.controller.Sessions', {
    extend: 'Ext.app.Controller',
    config: {
        refs: {
            loginForm: 'formpanel'
        },
        control: {
            'formpanel button': {
                tap: 'doLogin'
            }
        }
    },
    doLogin: function() {
        var form = this.getLoginForm(),
            values = form.getValues();
        MyApp.authenticate(values);
    }
});
```

doLogin 函数本身非常容易理解，由于前面定义了一个叫做 loginForm 的 ref，controller 将会自动生成一个叫做 getLoginForm 的函数用来返回该 formpanel，待完成对这个 form 的引用之后，只需要把其中的值（用户名和密码）取出来然后传递给身份验证函数即可。

7.4.2 数据存储器

Store（数据存储器）是 ST 的重要组成部分，它能够实现大部分的组件数据绑定工作。简单来说，一个 Store 就是一个由 Model（数据模型）的实例组成的数组，诸如 List 和 DataView 这类的数据绑定型控件，它们会为 Store 中的每一个 Model 实例渲染一个 item（这里指数据绑定控件的子项），Store 中的 Model 实例被添加或者删除的时候会触发数据绑定控件的相应事件，从而实现控件的更新。

7.4.3 设备配置文件

ST 可以跨越非常广泛的平台，尽管这些平台拥有不同的性能和屏幕尺寸。一个在平板电脑上工作良好的 UI 并不一定适应手机界面，反之亦然。所以为不同设备提供定制过的不同 view（视图）是一件很有必要的事情，同时又希望可以让不同设备共享尽可能多的代码。

Device Profile（设备配置）是一些简单的类，这些类能定义程序支持的不同类型设备以及如何处理这些不同。Device Profile 不是必需的，一开始不定义 Profile 以后再添加，甚至永远不定义它们。每个 Profile 都要定义一个简单的 isActive 函数，用来返回当前设备上是否应该启用此 Profile（换言之，该 Profile 是否匹配当前的设备），并为该 Profile 载入一堆（当前 Profile 中约定的）model、view 和 controller。

要为应用程序添加 Profile 支持功能，只需告诉应用程序有哪些 Profile 需要被支持，然后为它们各自创建 Ext.app.Profile 的子类即可。

```
Ext.application({
    name: 'MyApp',
    profiles: ['Phone', 'Tablet']
});
```

如上面代码所示，应用程序会加载 app/profile/Phone.js 和 app/profile/Tablet.js 两个文件，假定平板电脑的版本将会拥有一些额外的能力，比如对组的管理功能，下面的例子将演示如何定义 Tablet 的 Profile。

```
Ext.define('MyApp.profile.Tablet', {
    extend: 'Ext.app.Profile',
    config: {
        controllers: ['Groups'],
        views: ['GroupAdmin'],
        models: ['MyApp.model.Group']
    },
    isActive: function() {
        return Ext.os.is.Tablet;
```

```
    }
});
```


当 ST 检测到设备是一台平板电脑时，isActive 函数将会返回 true。鉴于现在不断涌现出的各种新设备，其外形和尺寸已经越来越模糊了手机和平板电脑的界限，故而无法界定哪些设备是平板哪些设备是手机，ST 的 Ext. os. is. Tablet 只有在 iPad 上运行的时候将被设定为 true，其他则为 false，如果需要更好的判断和控制，可以通过在 isActive 函数中进行更多的检测，来控制它返回 true 还是 false。

必须保证只有一个 Profile 的 isActive 函数可以返回 true，如果超过一个的话，只有第一个会有效而其他将被忽略，第一个返回 true 的 Profile 将被视做应用程序的当前 Profile。

如果检测到的当前 Profile 定义了额外的 model（数据模型）、view（视图）、controller（控制器）、store（数据存储器），它们会与 application 当中定义的其他元素一起被自动加载。而所有在 Profile 中定义的元素路径前面都会被加上 Profile 的名称，除非对它们定义了完整路径的类名，如下所示。

views：['GroupAdmin']将会加载 app/view/tablet/GroupAdmin. js（因为 GroupAdmin 是在 Tablet Profile 中配置的）。

controllers：['Groups']将会加载 app/controller/tablet/Groups. js（同上）。

models：['MyApp. model. Group']将会加载 app/model/Group. js（因为使用了完整路径）。

绝大多数情况下，Profile 只会定义额外的 controller 和 view，因为 model 和 store 一般情况下都会被共享。

7.4.4 应用启动

每个 Application 对象都会定义一个 launch 函数，它将会在应用程序所需的全部 class 加载完成，且应用程序已经做好准备的情况下执行。一般来说，这里就是用来放置应用程序启动逻辑的最好位置了，比如可以在这里为应用创建主要 view 框架。

除了 Application 中的 launch 函数之外，还有两个地方可以放置启动逻辑：①每个 controller（控制器）都可以定义一个 init 函数，这个函数将会运行在 application 的 launch 运行之前；②如果使用了设备 Profile，每一个 Profile 都可以定义一个 launch 函数，它将会在 controller 的 init 之后和 application 的 launch 之前被调用。

注意：只有活动 Profile 的 launch 函数才会被调用，比如分别定义了 phone 和 Tablet 的 Profile，现在是在 tablet 上运行它，那么只有 Tablet Profile 中的 launch 函数会被调用到。以下是调用顺序。

(1) Controller 的 init 首先被调用。

(2) 其次是当前 Profile 的 launch 被调用。

(3) 然后 Application 的 launch 被调用。

(4) 最后是其他 controller 的 launch 被调用。

7.4.5 路由和访问历史支持

ST2 具有完整的路由和访问历史支持，SDK 中的好几个例子都使用了历史路径支持，

以实现通过后退按钮可以轻易地在屏幕之间回退导航，这一点在 Android 上尤其有用。

7.5 组件的使用

ST 有着自己丰富的组件，在 ST 中接触的很多类都是组件，每个组件都是 Ext. Component 的子类。使用这些组件可以构建绝大多数应用程序，如常用的表单组件、面板组件、按钮组件等，下面对这些组件做一个详细的分类，见表 7.1。

表 7.1 ST 组件

组件名	类	xtype	备注
Component	Ext. Component	无	所有组件的父类
Audio	Ext. Audio	audio	音频组件
Button	Ext. Button	button	按钮组件
Label	Ext. Label	label	标签组件
Img	Ext. Img	img 或者 image	图片组件
Mask	Ext. Mask	mask	加载标记组件
LoadMask	Ext. LoadMask	loadmask	加载标记组件
Map	Ext. Map	map	地图组件
MessageBox	Ext. MessageBox	无	弹出消息框
Msg	Ext. Msg	无	弹出消息框
SegmentedButton	Ext. SegmentedButton	segmentedbutton	按钮
Spacer	Ext. Spacer	spacer	分隔线
Toolbar	Ext. Toolbar	toolbar	工具条
TitleBar	Ext. TitleBar	titleBar	标题栏工具
Viewport	Ext. Viewport	无	视图组件
Video	Ext. Video	video	视频组件
DatePicker	Ext. picker. Date	datepicker	下拉框日历选择器
Picker	Ext. picker. Picker	picker	通用下拉框组件
Panel	Ext. tab. Panel	panel	面板
FieldSet	Ext. form. FieldSet	fieldset	表单组件
ActionSheet	Ext. ActionSheet	actionsheet	自下而上弹出式按钮组，iOS 设备上常用
Carousel	Ext. carousel. Carousel	carousel	滑动面板组件，在多个面板之间通过滑动进行切换
DataView	Ext. dataview. DataView	dataview	数据视图组件，常用于 list 组件
IndexBar	Ext. dataview. IndexBar	indexbar	快速检索工具条，常见于智能移动设备通讯录的快速检索工具条
List	Ext. dataview. List	list	数据列表组件，类似于表格控件
NestedList	Ext. dataview. NestedList	nestedlist	数据列表组件，不同于 list 组件的是，它提供树形展现方式，根据树的层级结构展示列表内容
View	Ext. navigation. View	navigationview	导航视图组件，可以通过 push 和 pop 自动管理组件，任意时刻只有一个活动的视图

7.5.1 容器

应用程序是由很多组件组成的，它们被一个个的组件包含着。容器也像组件，但除了组件的功能以外，还可以渲染和插入新的组件。大部分 App 都有唯一的一个最上层容器叫做“Viewport”，它占满了整个屏幕，子组件被包含在它们里面。容器主要有三个功能：在初始化和运行的时候添加新的组件；移除组件；指定组件布局。

布局确定了组件在屏幕上的显示方式，ST 提供了多种布局方式，可以方便开发者完成组件布局。

7.5.2 初始化组件

组件的初始化和 ST 中其他类的初始化一样，使用 Ext. create 方法。

7.5.3 配置组件

用户可以通过 configuration 选项对任意一个组件进行配置。所有的 configuration 都在组件类的 Config Options 中。可以在组件初始化时传入任意个配置选项，也可以在组件初始化之后对它的配置进行修改。

任何一个配置都有 Getter 和 Setter 方法，它们是自动生成的。例如，一个配置选项叫做“html”，那么将会有一个 getHtml 和一个 setHtml 方法；一个默认的配置拥有一个 getDefault 方法和一个 setDefault 方法。

7.5.4 使用 ST 已有的组件

ST 中有两种方式用来创建组件，第一种是在 config 中通过 xtype 直接申明组件，第二种是通过 Ext. create()方法创建组件。

```
Ext.define('NoticeApp.view.LoginViewport',{
    extend:'Ext.Panel',
    xtype:'loginviewport',
    requires: [
        'NoticeApp.view.SystemLogin'
    ],
    config:{
        items:[
        {
            id:'loginNavigationview',
            xtype:'navigationview',
            navigationBar:false,
            items:[
                {
                    xtype:'systemlogin'
                }
            ]
        }
        ]
```

```
        },
        initialize: function(){
            //to do something
        }
});
```

如上面示例代码中，通过 xtype：navigationview 申明了一个 navigationview 导航视图组件。

通过 xtype：systemlogin 申明了一个组件，请注意 systemlogin 这个组件或者说是视图在标准的 ST 组件中并没有找到，是因为这是一个自定义的组件或者说是视图。

下面是通过 Ext. create 方法创建组件

```
Ext.create('NoticeApp.view.MainViewport',{
                fullscreen: true,
                layout: 'card'
});
```

这段示例代码中通过 Ext. create 方法创建了一个类为 NoticeApp. view. MainViewport 的组件，同样，MainViewport 这个视图是自定义的视图，通常在 ST 开发中，需要去继承已有的组件类来扩展或者自定义自己的视图，这样符合 MVC 的开发模式，通过将各种组件通过一定的布局方式黏合在一起形成自己的视图组件，也为实现组件复用奠定了基础。

7.5.5 定义自己的组件（视图类）

ST 提供的组件虽然有很多，但是这些定制化的组件样式、包括布局等并不一定都能满足用户所有的要求，因此有时候需要自己定义自己的组件，这当然是在继承已有的组件基础之上的。在 ST 中通过 Ext. define 方法类定义自己的类，这个类可以继承现有的 ST 类，也可以是自己独立开发的一个类。在定义类的同时，可以给这个类注册一个 xtype，注册 xtype 有什么好处呢？一类注册 xtype 后，任何地方只要加入对此类的依赖，通过 xtype 就可以使用该类或者说是组件、视图。比如说，通过继承 MessageBox 类来改变 ST 默认的消息弹出框风格，这样可以指定一个 xtype，今后就可以直接使用这个 xtype 类，使用自定义风格的消息弹出框了。当然也可以通过 Ext. create 通过类的全限定名来创建该类。

7.6 布　　局

7.6.1 Box 布局

Sencha Touch 里的布局有 5 种：分别是 hbox、vbox、card、fit、auto（默认）。

实际上可以分为 Box 布局和 Fit 布局两种。Sencha Touch 里的布局应该理解为：该控件内部子项的排列方式。先来看看 Box 布局。

顾名思义，Box 布局就是一个个的 Box 组成的，它又分为 hbox 和 vbox。hbox 是水平排列、垂直居中、靠左置顶；vbox 是竖直堆叠、水平居中、靠上置顶。

hbox 布局，如图 7.7 所示。

vbox 布局，如图 7.8 所示。



图 7.7　hbox 布局　　　　图 7.8　vbox 布局

但是，只是知道这些，还不足以清楚 Box 布局，下面看看其他常见的 Box 布局实例。vbox 变型一，如图 7.9 所示。

vbox 变型二，在代码中加入 width："100%"，如图 7.10 所示。



图 7.9　vbox 变型一



图 7.10　vbox 变型二

了解以上内容之后，再来看看经典的九宫格布局的实现，经典的九宫格布局，如图 7.11 所示。

在经典的九宫格布局的基础上设置 margin、padding 属性，就成为松散九宫格布局，如图 7.12 所示。



图 7.11　经典的九宫格布局



图 7.12　松散九宫格布局

到这里，对 Box 布局已经有了一个大致的了解，接下来将接着讲解 Sencha Touch 里的另一种布局。

7.6.2 Card 布局

1. Fit 布局

Fit 布局很有特点，它只允许自己的第一个 item 被显示出来，并且填充满自己。如果它的 item 多于一个，难免会出一些意外的情况。

如果发现容器内的控件有的没有全部被显示出来，那不妨看看它的 layout 是否为 fit，试着将它强制设为 vbox 往往就能解决问题。

图 7.13 是在一个 layout 为 fit 的 Panel 里放了三个 button 的效果。

所以说，除非应用程序需要这种效果，否则请不要在 fit 布局的容器里面放超过一个对象。



图 7.13 Fit 布局

2. Card 布局

Sencha Touch 中，Card 布局继承 Fit 布局，也是使用得比较多的一个布局，许多的界面切换都需要这个布局的 Panel 的参与。Sencha Touch 中使用 Card 布局的有：TabPanel、Carousel。Card 布局内的东西就是把它一层层堆在里面，要显示哪一层的时候，只需要调用容器的 setActiveItem。

另外，因为 TabPanel 、Carousel 本身都是 Card 布局，因此可以调用它的 setActiveItem()来改变当前显示的内容。

下面的例子演示了如何使用 Card 布局进行画面的切换，代码如下。

```
var myApp = new Ext.Application({
    name: 'myApp',
    launch: function () {

        var myPanel1 = new Ext.Panel({
            id: 'myPanel1',
            layout: 'vbox',
            html: 'Oh, this is Panel1! ',
            items: {
                xtype: 'button',
                text: '前往 Panel2',
                handler: function(){
                    myApp.views.mainPanel.setActiveItem(//设置活动项的方法
                        'myPanel2',                    //第一个参数为 myPanel2 的 id
                        'slide'                        //这个参数为切换效果
                    );
                }
            }
        });

        var myPanel2 = new Ext.Panel({
            id: 'myPanel2',
            layout: 'vbox',
```

```
                html:  'This is Panel2!',
                items: {
                    xtype: 'button',
                    text: '前往 Panel3',
                    handler: function(){
                        var pnl  = new Ext.Panel({
                            html: '点击按钮之后才创建的 Panel,演示到此结束'
                        });

                        myApp.views.mainPanel.setActiveItem(
                            pnl,{                       //这个参数是刚创建的 panel
                            type: 'slide',              //这个参数为一个动画效果对象
                            direction: 'right'
                        });
                    }
                }
            });

            myApp.views.mainPanel  = new Ext.Panel({
                fullscreen: true,
                layout: 'card',
                items: [myPanel1, myPanel2]             //第一个为默认界面
            });
        }
    });
```

小　　结

本章主要介绍了 Sencha Touch 如何搭建开发环境,其开发模式,以及组件和布局,通过本章的学习,可以利用 Sencha Touch 的特性去开发 Web 应用程序。

第 8 章 PhoneGap 应用

本章学习目标

- PhoneGap 开发环境搭建
- PhoneGap 的基本使用

PhoneGap 是一个用基于 HTML、CSS 和 JavaScript 的，用来创建跨平台移动应用程序的快速开发框架。它使开发者能够利用 iPhone、Android、Palm、Symbian、WP7、Bada 和 Blackberry 智能手机的核心功能——包括地理定位、加速器、联系人、声音和振动等，此外 PhoneGap 拥有丰富的插件，可以以此扩展无限的功能。PhoneGap 是免费的，但是它需要特定平台提供的附加软件，例如 iPhone 的 iPhone SDK，Android 的 Android SDK 等，也可以和 DW5.5 配套开发。使用 PhoneGap 只比为每个平台分别建立应用程序好一点点，因为虽然基本代码是一样的，但是仍然需要为每个平台分别编译应用程序。

PhoneGap 针对不同平台的 WebView 做了扩展和封装，使 WebView 这个组件变成可访问设备本地 API 的强大浏览器，所以开发人员在 PhoneGap 框架下可通过 JavaScript 访问设备本地 API。WebView 是什么？WebView 组件实质是移动设备的内置浏览器，WebView 这个内置浏览器的特性是 Web 能被打包成本地客户端的基础，可方便地用 HTML5、CSS3 页面布局。

PhoneGap 的优势在于以下几点。

(1) 可跨平台：PhoneGap 是唯一支持 7 个平台的开源移动框架(PhoneGap 包括地理定位、加速器、联系人、声音和振动等，此外 PhoneGap 拥有丰富的插件，可以以此扩展无限的功能，几乎 Native App 能完成的功能它都能完成)。它的优势是无与伦比的：开发成本低——据估算，至多是 Native App 的五分之一。

(2) 易用性，基于标准的 Web 开发技术(HTML+CSS+JavaScript)。

(3) 提供硬件访问控制(API)。

(4) 可利用成熟 JavaScript 框架(JQueryMobile SenchaTouch)。

(5) 安装和使用方便。

随着 2011 年 10 月 4 日 Adobe 宣布收购了创建了 HTML5 移动应用框架 PhoneGap 和 PhoneGap Build 的新创公司 Nitobi Software。这使得 PhoneGap 有了坚强的后盾，PhoneGap 的发展前景也是一片光明。与此同时，PhoneGap 的开源框架已经被累计下载 60 万次，借助 PhoneGap 平台，已有数千应用程序建立在 iOS、Android 以及其他操作系统之上。

8.1　开发环境搭建(Android 平台)

PhoneGap 是一个开发跨平台的 HTML5 本地化程序的平台,通过它可以把网页变为各种平台上的应用程序。开发环境的搭建主要有以下 4 个步骤。

(1) 到官网 http://phonegap.com 下载 PhoneGap。笔者下载的版本是 PhoneGap-2.9.0。

(2) 浏览下载完的 Zip 包,如图 8.1 所示。



图 8.1　PhoneGap Zip 包内容

(3) 在系统里已经安装好了 Eclipse 3.8 和 Android 4.0。

(4) 新建工程,如图 8.2 所示。

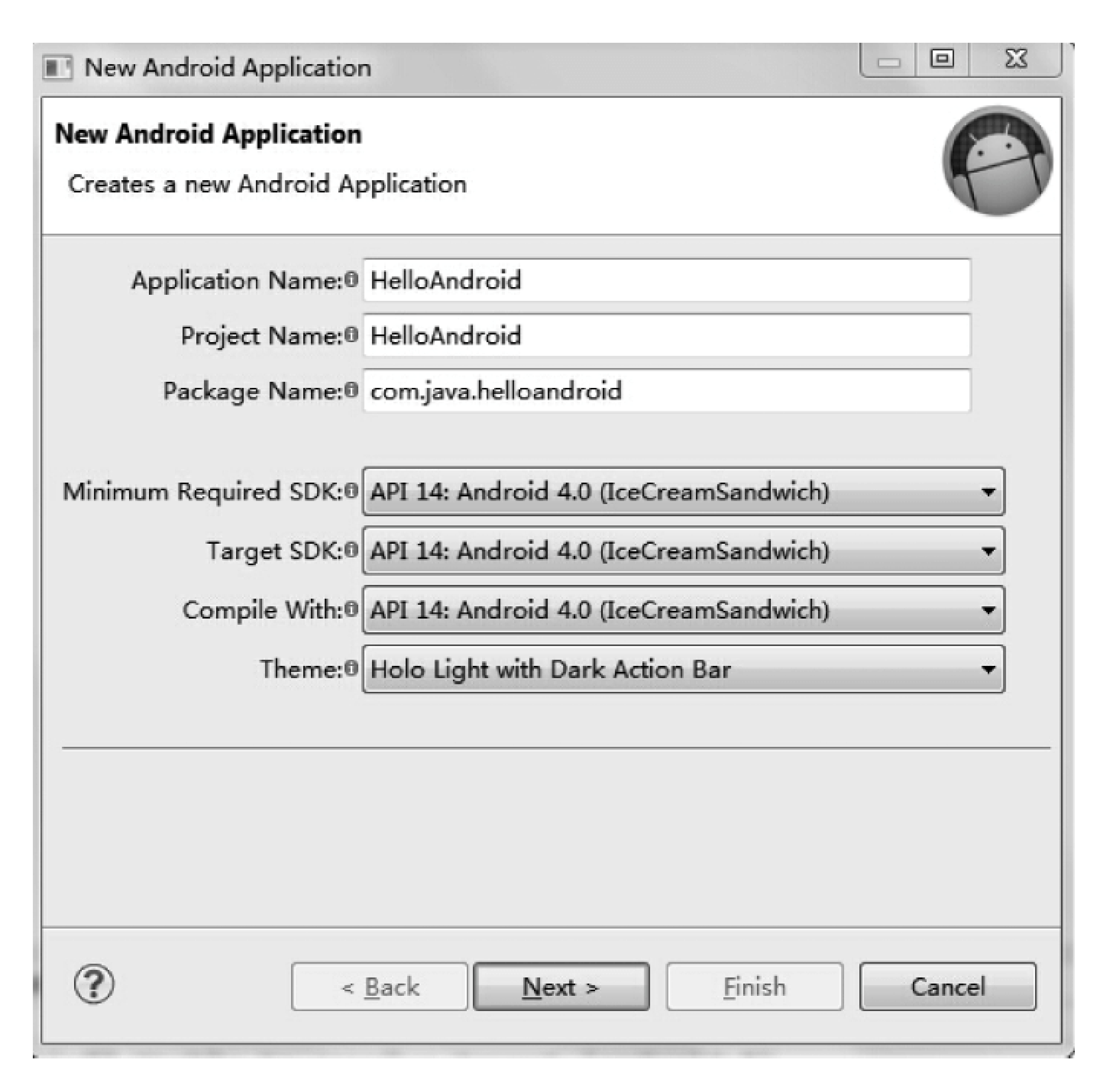


图 8.2　新建工程

（5）加入 PhoneGap 包。

① 创建目录：

/libs 目录：这个用于放 cordova-2.9.0.jar。

/assets/www：这个用于放 cordova.js，如图 8.3 所示。

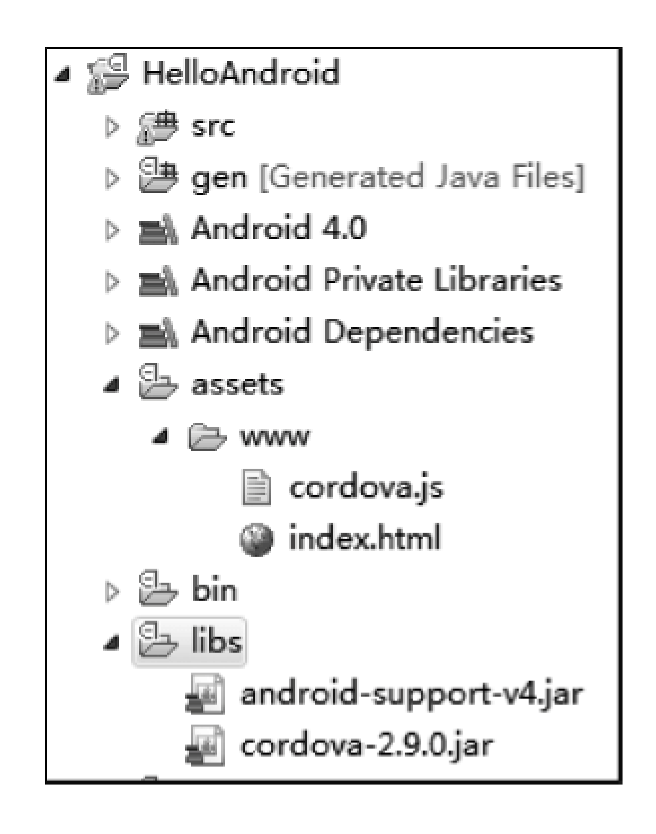


图 8.3　创建目录

② 单击项目右键属性→Java Build Path→Libraries→Add Jars，如图 8.4 和图 8.5 所示。

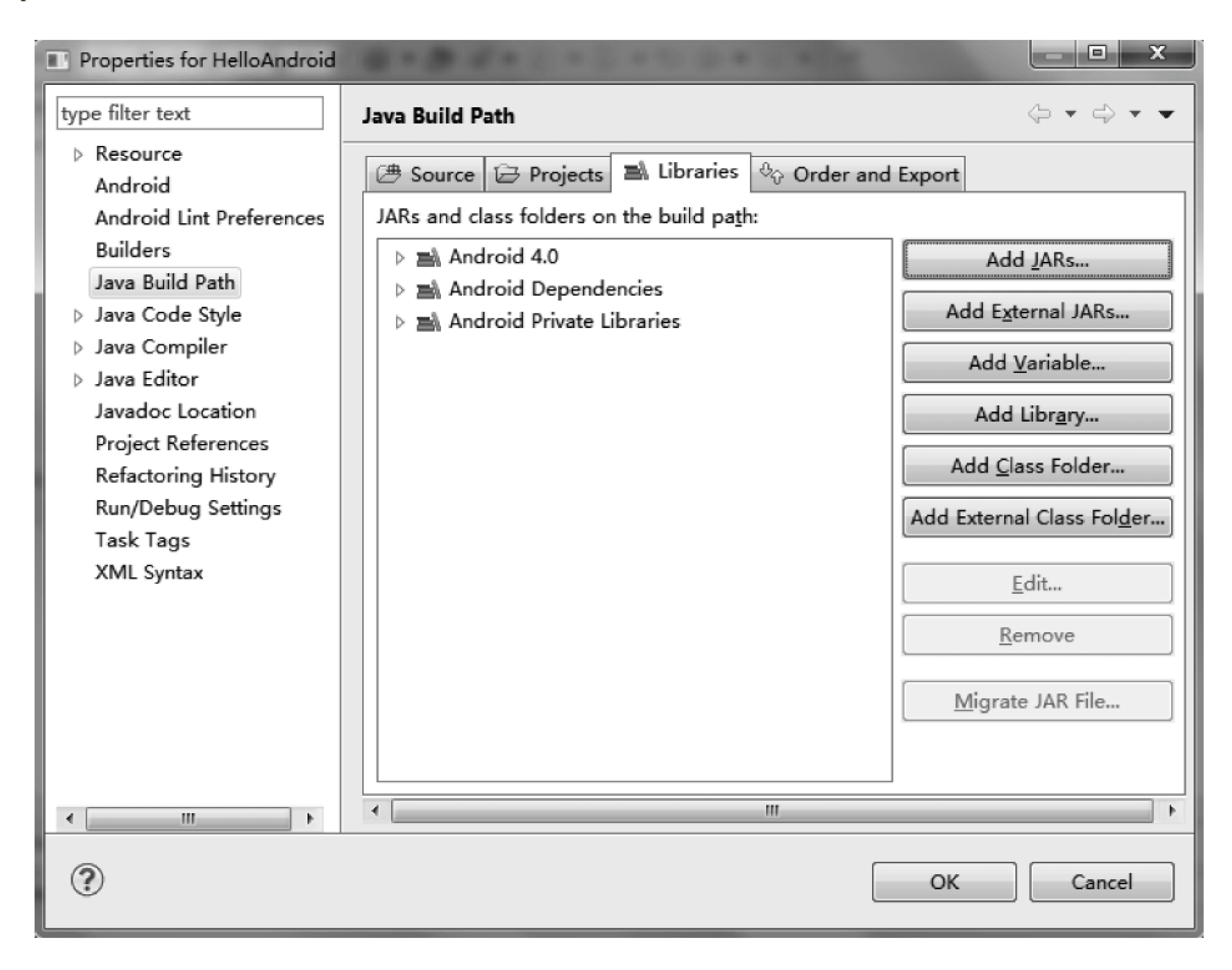


图 8.4　加入 PhoneGap 包

③ 把解压出来的 android 目录下的 xml 目录放到/res 目录下，如图 8.6 所示。

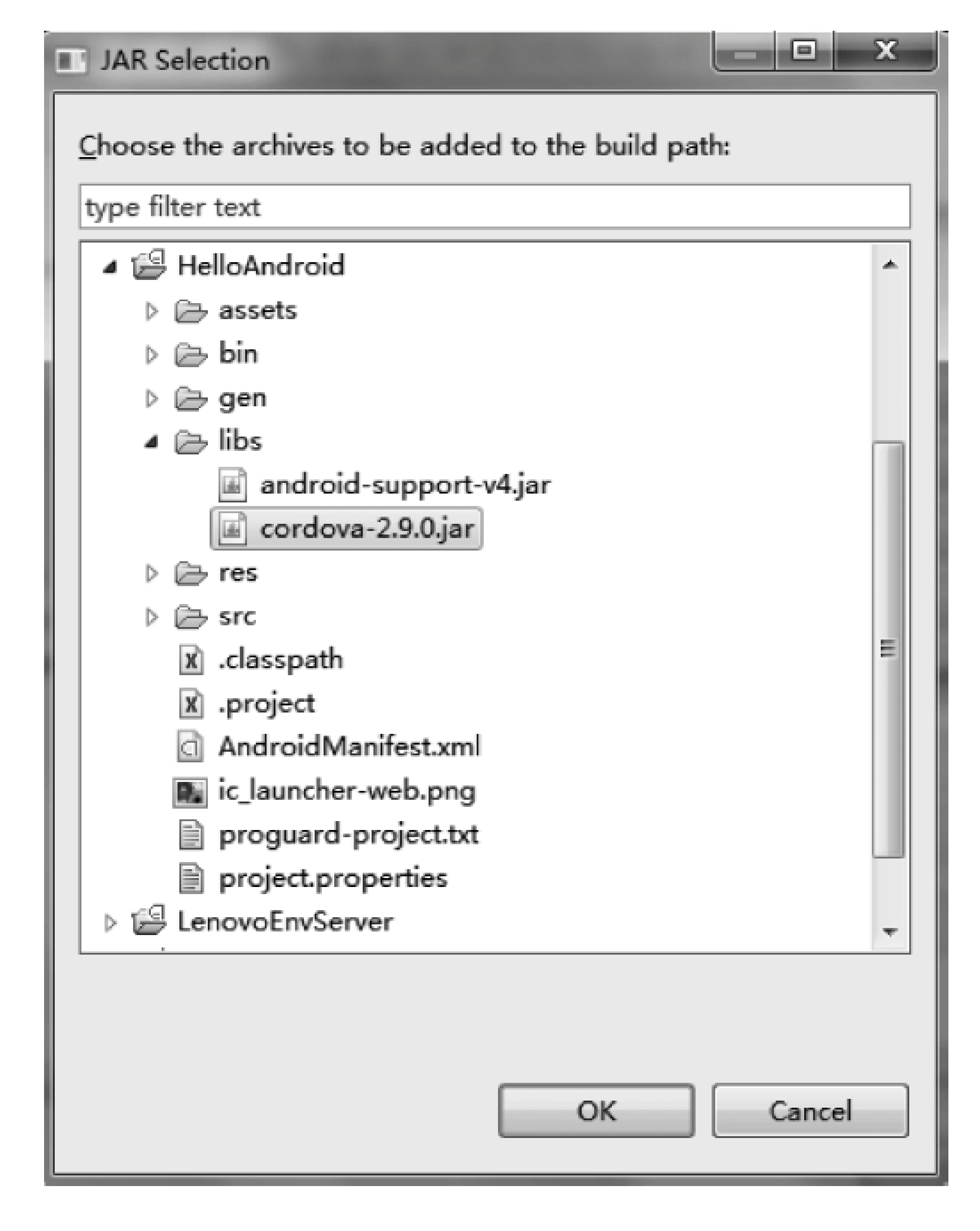


图 8.5　加入 PhoneGap 包

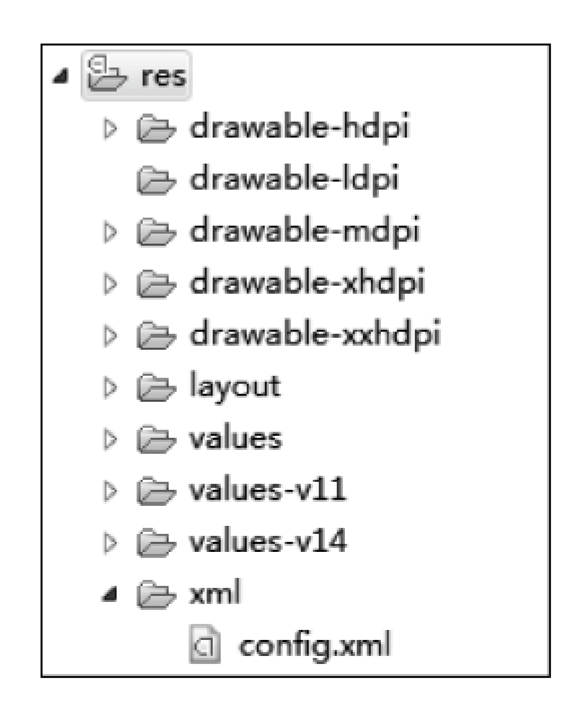


图 8.6　xml 目录放到/res 目录下

(6) 开始开发,可以参考自带的例子。

① 首先把 activity 类进行修改。

```
public class HelloPhoneGap extends DroidGap {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        //加载 assets/www 目录下的 index.html
        super.loadUrl("file:///android_asset/www/index.html");
    }
}
```

② 创建 index.html,编辑 assets/www/index.html。

```
<!DOCTYPE HTML>
<head>
 <title>你好,PhoneGap</title>
 <script type="text/javascript" charset="utf-8" src="cordova-2.9.0.js">
 </script>
</head>
<body>
    <h1>你好,PhoneGap</h1>
</body>
</html>
```

③ 修改 AndroidManifest.xml,增加权限设置。

```
<supports-screens android:largeScreens="true"
                  android:normalScreens="true"
                  android:smallScreens="true"
                  android:resizeable="true"
                  android:anyDensity="true"/>
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.RECORD_VIDEO"/>
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.BROADCAST_STICKY" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

(7) 运行程序。

① 创建模拟设备,如图 8.7 所示。

② 运行,如图 8.8 所示。

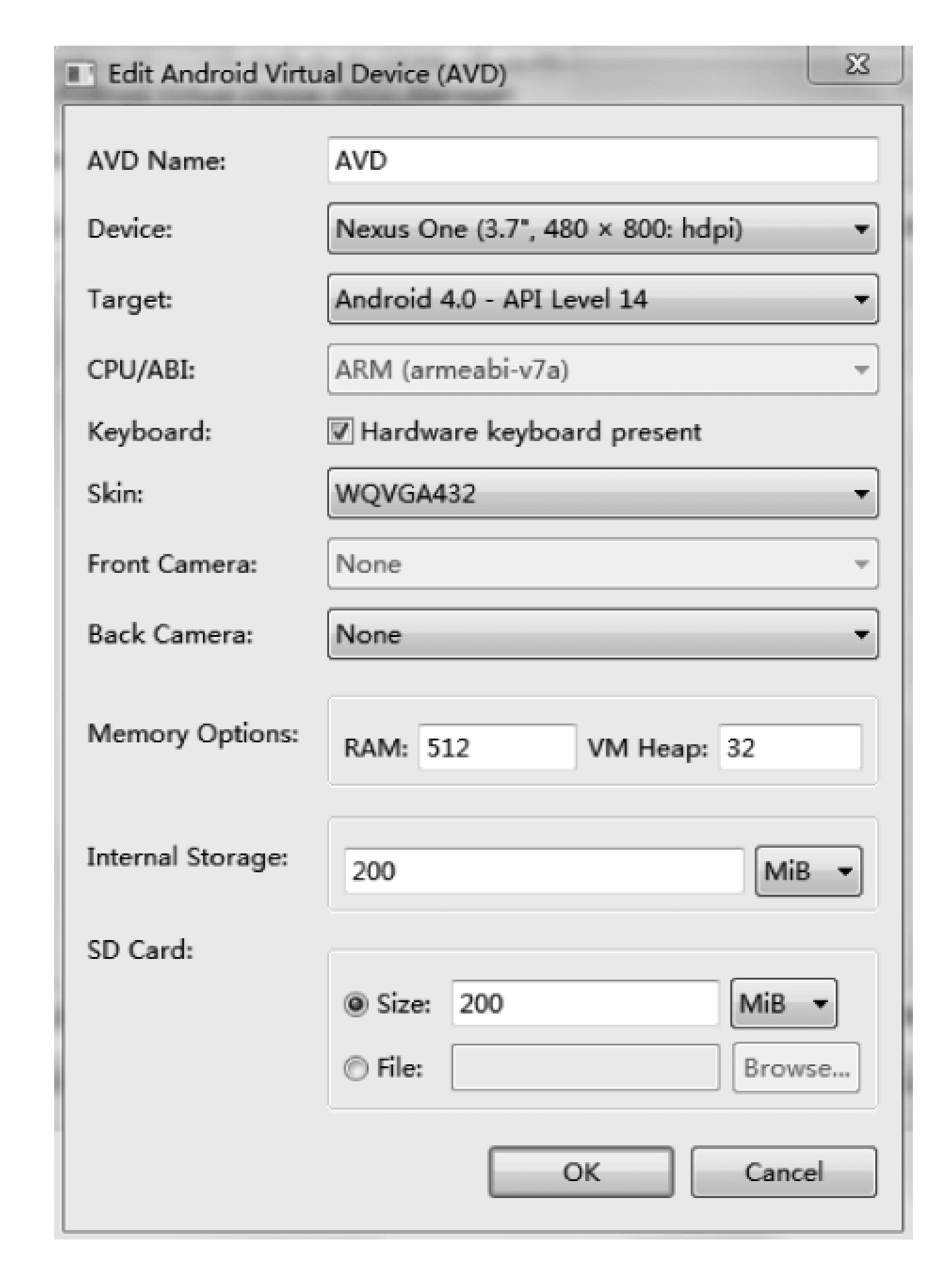


图 8.7 创建模拟设备

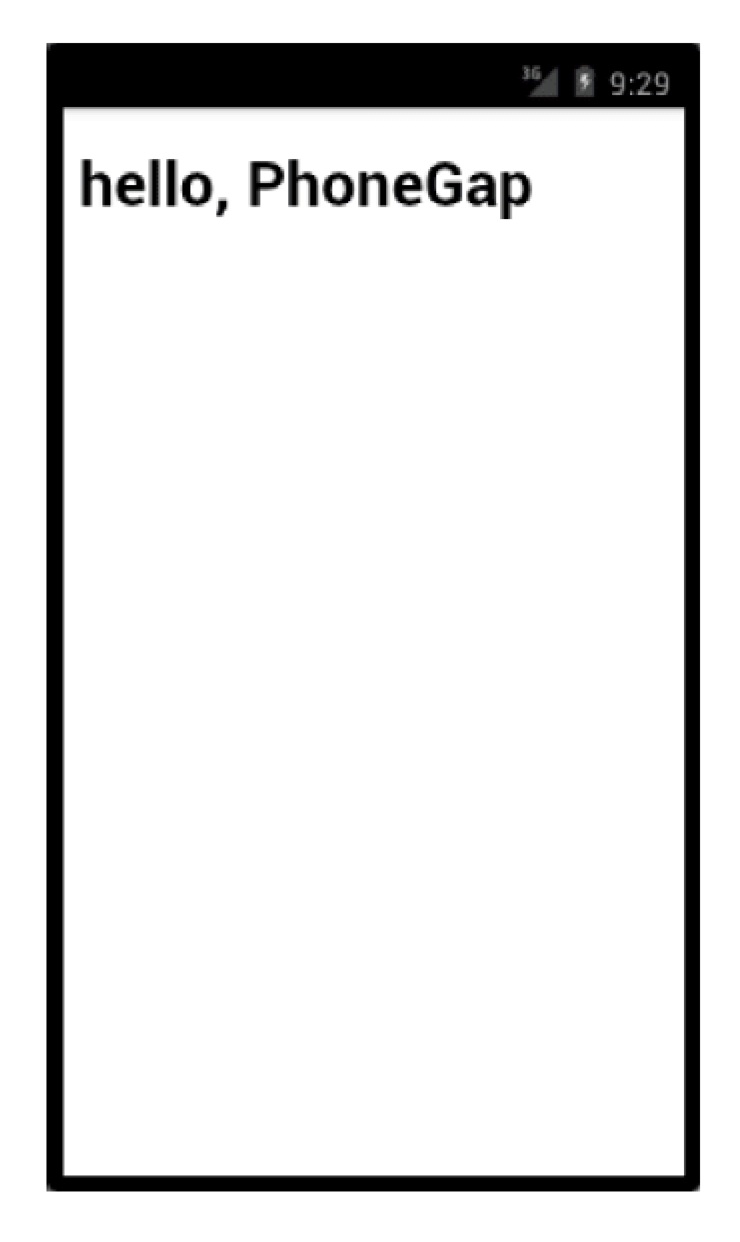


图 8.8 程序运行

8.2 Acceleration

Accelerometer 主要作用是采集设备在 x、y、z 方向上的动作。也就是返回在 x、y、z 方向上的加速，支持的平台：Android、BlackBerry WebWorks(OS 5.0 或更高版本)、iOS。Accelerometer 实现这个功能有以下三个方法：

(1) accelerometer.getCurrentAcceleration

获取其在 xyz 方向的加速度。

(2) accelerometer.watchAcceleration

对比第(1)个方法，这个可以每隔一段时间获取一次 xyz 方向的加速度。

(3) accelerometer.clearWatch

停止对第(2)个方法定义的加速器的监视。

下面给出一段示例代码：

```
<html>
<head>
<title>Acceleration Example</title>
<script type="text/javascript" charset="utf-8" src="PhoneGap.js"></script>
<script type="text/javascript" charset="utf-8">
    //等待加载 PhoneGap
    document.addEventListener("deviceready", onDeviceReady, false);
    //PhoneGap 加载完毕
    function onDeviceReady() {
       navigator.accelerometer.getCurrentAcceleration(onSuccess, onError);
    }
    //onSuccess: 返回当前加速度数据的快照
    function onSuccess(acceleration) {
        alert('Acceleration X: ' + acceleration.x + '\n' +
            'Acceleration Y: ' + acceleration.y + '\n' +
            'Acceleration Z: ' + acceleration.z + '\n' +
            'Timestamp: '      + acceleration.timestamp + '\n');
    }

    //onError: 返回加速度数据失败
    function onError() {
        alert('onError!');
    }
</script>
</head>
<body>
    <h1>Example</h1>
    <p>getCurrentAcceleration</p>
</body>
</html>
```

8.3　Camera

当使用设备的摄像头采集照片的时候，Camera 对象提供对设备默认摄像头应用程序的访问。它支持的平台有：Android、BlackBerry WebWorks(OS 5.0 或更高版本)、iOS。它主要的参数和方法如下。

(1) 参数

- cameraSuccess——提供图像数据的 onSuccess 回调函数。
- cameraError——提供错误信息的 onError 回调函数。
- cameraOptions——定制摄像头设置的可选参数，它主要有以下参数：
  ✍quality：存储图像的质量，范围是[0,100]。
  ✍destinationType：选择返回数据的格式。
  ✍sourceType：设定图片来源。
  ✍allowEdit：在选择图片进行操作之前允许对其进行简单编辑(只有 iOS 支持)。
  ✍encodingType：选择返回图像文件的编码方式。
  ✍targetWidth：以像素为单位的图像缩放宽度指定图片展示的时候的宽度。
  ✍targetHeight：以像素为单位的图像缩放高度指定图片展示的时候的高度。
  ✍saveToPhotoAlbum：拍完照片后是否将图像保存在设备上的相册中。
  ✍mediaType：设置选择媒体的类型。
  ✍cameraDirection：选择前置摄像头还是后置摄像头。

(2) 方法

- camera.getPicture。

该方法会打开设备的默认摄像头应用程序，使用户可以拍照(如果 Camera.sourceType 设置为 Camera.PictureSourceType.CAMERA，这也是默认值)。拍照结束后，摄像头应用程序会关闭并恢复用户的应用程序。

如果 Camera.sourceType = Camera.PictureSourceType.PHOTOLIBRARY，或者是 Camera.sourceType=Camera.PictureSourceType.SAVEDPHOTOALBUM，系统弹出照片选择对话框，用户可以从相集中选择照片。返回值会按照用户通过 cameraOptions 参数所设定的下列格式之一发送给 cameraSuccess 回调函数：

✍一个字符串，包含 Base64 编码的照片图像(默认情况)。
✍一个字符串，表示在本地存储的图像文件位置。

该方法简单的范例如下：

```
navigator.camera.getPicture(cameraSuccess,cameraError,[cameraOptions]);
```

- camera.cleanup：清空使用摄像头拍照时候产生的缓存文件。

当使用如下参数的时候会产生临时文件，这时候使用这个方法会及时清除临时文件。使用方法如下：

```
navigator.camera.cleanup(cameraSuccess,cameraError);
```

这个方法的回调方法中没有参数，只是去调用对应的方法，可以在对应的方法中判断是否清空了。

再看下面一个简单的例子：

```
<!DOCTYPE html>
<html>
<head>
<title>Capture Photo</title>
<script type="text/javascript" charset="utf-8" src="js/cordova-2.6.0.js">
</script>
<script type="text/javascript" charset="utf-8">
    var pictureSource;          //设定图片来源
    var destinationType;        //选择返回数据的格式
    document.addEventListener("deviceready",onDeviceReady,false);

        function onDeviceReady(){
            pictureSource=navigator.camera.PictureSourceType;
            destinationType=navigator.camera.DestinationType;
        }

        function onPhotoDataSuccess(imageData) {
          //base64 encoded image data
          var smallImage = document.getElementById("smallImage");
          smallImage.style.display = "block";
         //在使用 base64 编码的时候需要使用这样的前缀
          smallImage.src = "data:image/jpeg;base64," + imageData;
        }

        //Called when a photo is successfully retrieved
        function onPhotoURISuccess(imageURI) {
          //image file URI
          var largeImage = document.getElementById("largeImage");
          largeImage.style.display = "block";
         //使用 image file URI 直接赋值就可以了
          largeImage.src = imageURI;
        }

        //第一个按钮调用函数
        function capturePhoto() {
          navigator.camera.getPicture(onPhotoDataSuccess, onFail, { quality: 50,
            destinationType: destinationType.DATA_URL });
        }

        //第二个按钮调用的函数
        function capturePhotoEdit() {
          navigator.camera.getPicture(onPhotoDataSuccess, onFail,{quality: 20, allowEdit: true,
            destinationType: destinationType.DATA_URL });
        }

        //第三/四个按钮调用的函数
        function getPhoto(source) {
          //Retrieve image file location from specified source
```

```
            navigator.camera.getPicture(onPhotoURISuccess, onFail, { quality: 50,
              destinationType: destinationType.FILE_URI,
              sourceType: source });
          }

          function onFail(message) {
            alert("Failed because: " + message);
          }

          </script>
        </head>
        <body>
        <button onclick = "capturePhoto();">Capture Photo</button><br>
        <button onclick = "capturePhotoEdit();">Capture Editable Photo</button><br>
        <button onclick = "getPhoto(pictureSource.PHOTOLIBRARY);">
         From Photo Library
        </button><br>
        <button onclick = "getPhoto(pictureSource.SAVEDPHOTOALBUM);">
          From Photo Album
        </button><br>
      <img style = "display:none;width:60px;height:60px;" id = "smallImage" src = "" />
      <img style = "display:none;" id = "largeImage" src = "" />
    </body>
  </html>
```

8.4 capture.captureAudio

Capture 主要是提供对设备音频、图像和视频采集功能的访问。其中 capture.captureAudio 方法是对音频设备的访问，也就是指录音功能。capture.captureAudio 方法启动录音机应用程序并返回采集的音频剪辑文件。该方法通过设备默认的音频录制应用程序开始一个异步操作以采集音频录制。该操作允许设备用户在一个会话中同时采集多个录音。当用户退出音频录制应用程序，或系统到达 CaptureAudioOptions 的 limit 参数所定义的最大录制数时都会停止采集操作。如果没有设置 limit 参数的值，则使用其默认值 1，也就是说当用户录制好一个音频剪辑后采集操作就会终止。

当采集操作结束后，系统会调用 CaptureCB 回调函数，传递一个包含所有采集到的音频剪辑文件的 MediaFile 对象数组。如果用户在完成一个音频剪辑采集之前终止采集操作，系统会调用 CaptureErrorCB 回调函数，并传递一个包含 CaptureError.CAPTURE_NO_MEDIA_FILES 错误代码的 CaptureError 对象。

简单例子代码如下：

```
<!DOCTYPE html>
<html>
  <head>
    <title>capture.captureAudio</title>
    <script type = "text/javascript" charset = "utf - 8" src = "js/cordova - 2.6.0.js">
    </script>
```

```
    <script type="text/javascript" charset="utf-8">
    //captureAudio 方法成功执行后回调函数
    function captureSuccess(mediaFiles) {
        var i, len;
        for (i = 0, len = mediaFiles.length; i < len; i += 1) {
           //业务逻辑
            navigator.notification.alert(mediaFiles[i].fullPath + " " + mediaFiles[i].name);
        }
    }

    //captureAudio 方法执行失败后回调函数
    function captureError(error) {
        var msg = "capture 发生错误: " + error.code;
        navigator.notification.alert(msg, null, "oh!");
    }

    function captureAudio() {
        //limit 录制的音频数
        navigator.device.capture.captureAudio(captureSuccess, captureError,
        {limit: 1});
    }

    </script>
  </head>
  <body>
   <h1>capture.captureAudio</h1><br/>
    <button onclick="captureAudio();">Capture Audio</button><br>
  </body>
</html>
```

按钮单击后的效果如图 8.9 所示。

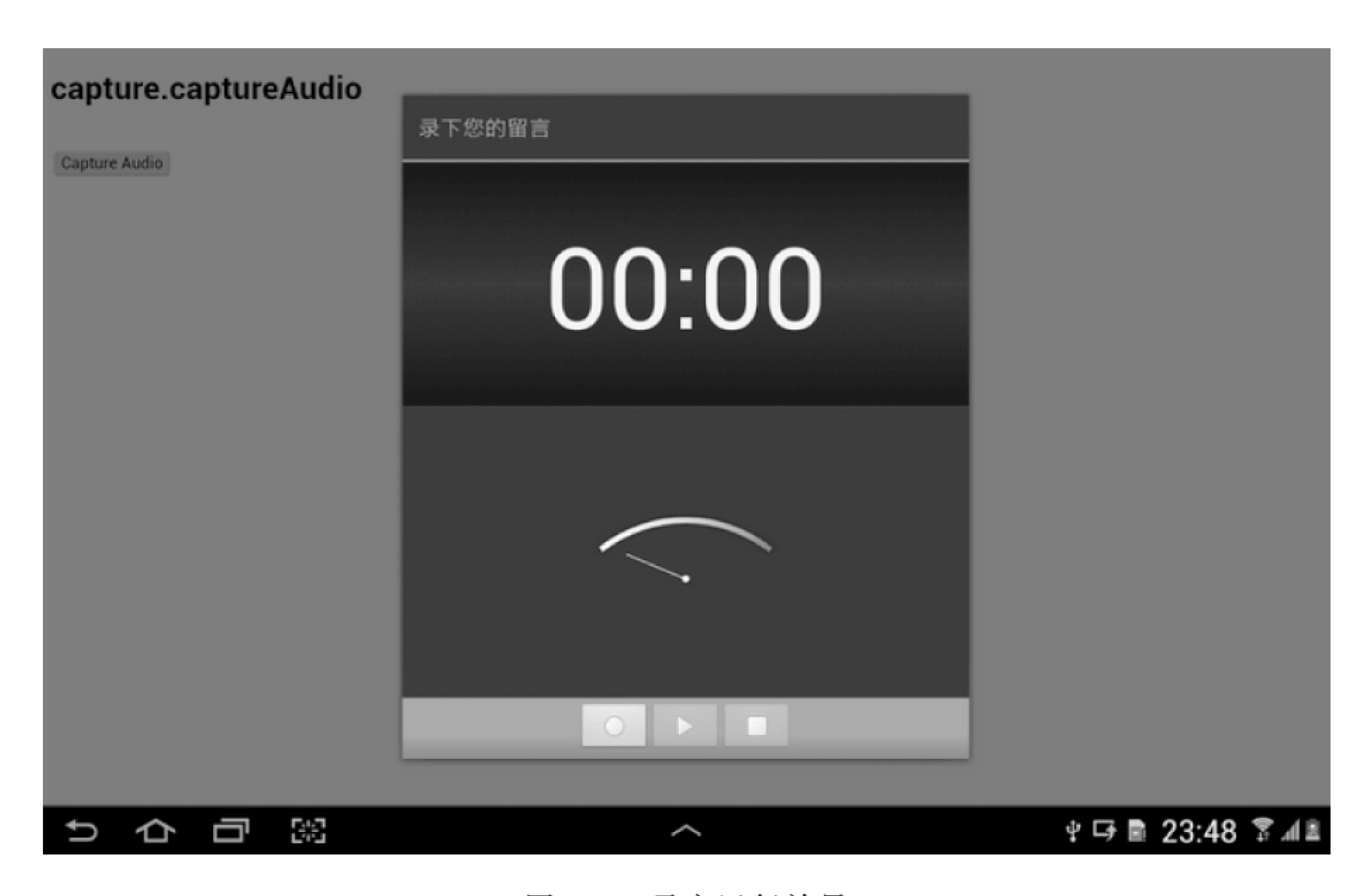


图 8.9　录音运行效果

录音结束后如图 8.10 所示。

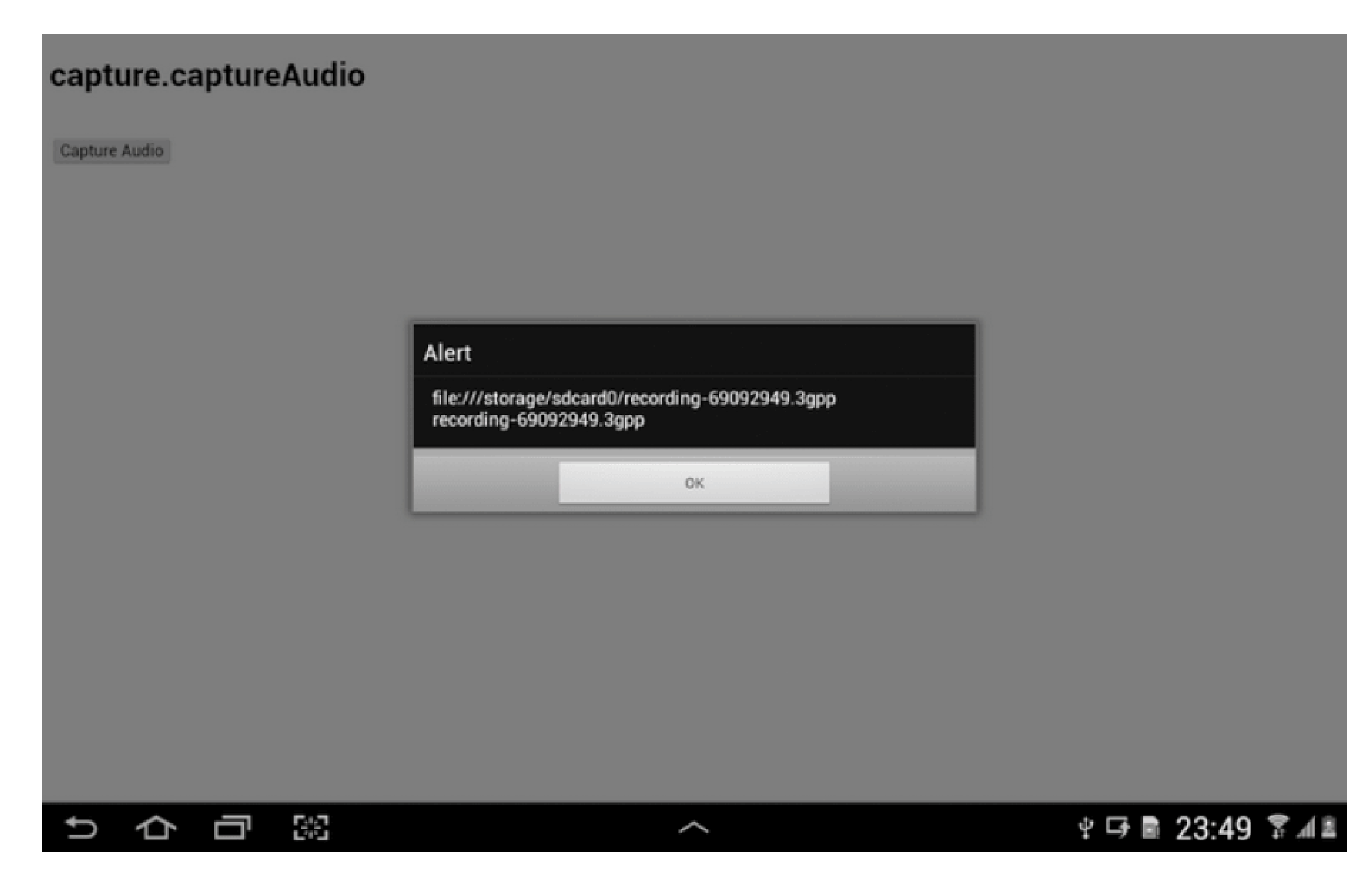


图 8.10　录音结束

8.5　capture.captureImage

capture.captureImage 方法开启摄像头应用程序，返回采集到的图像文件信息。该方法通过设备的摄像头应用程序开始一个异步操作以采集图像。该操作允许设备用户在一个会话中同时采集多个图像。

当用户退出摄像头应用程序，或系统到达 CaptureImageOptions 的 limit 参数所定义的最大图像数时都会停止采集操作。如果没有设置 limit 参数的值，则使用其默认值 1，也就是当用户采集到一个图像后采集操作就会终止。

当采集操作结束后，系统会调用 CaptureCB 回调函数，传递一个包含每个采集到的图像文件的 MediaFile 对象数组。如果用户在完成一个图像采集之前终止采集操作，系统会调用 CaptureErrorCB 回调函数，并传递一个包含 CaptureError.CAPTURE_NO_MEDIA_FILES 错误代码的 CaptureError 对象。

以下是示例代码。

```
<!DOCTYPE html>
<html>
  <head>
    <title>capture.captureAudio</title>
    <script type = "text/javascript" charset = "utf-8"
            src = "js/cordova-2.6.0.js">
    </script>
    <script type = "text/javascript" charset = "utf-8">
    //captureAudio 方法成功执行后回调函数
```

```
        function captureSuccess(mediaFiles) {
          var i, len;
          for (i = 0, len = mediaFiles.length; i < len; i += 1) {
            //业务逻辑
            navigator.notification.alert(mediaFiles[i].fullPath + mediaFiles[i].name);
          }
        }
        //captureAudio 方法执行失败后回调函数
        function captureError(error) {
            var msg = "capture 发生错误: " + error.code;
            navigator.notification.alert(msg, null, "oh");
        }

        function captureImage() {
          //limit 拍照的张数
          navigator.device.capture.captureImage(captureSuccess,captureError,
          {limit:2});
        }
        </script>
      </head>
      <body>
       <h1>capture.captureImage</h1><br/>
        <button onclick="captureImage();">Capture Audio</button><br>
      </body>
    </html>
```

运行后如图 8.11 所示。

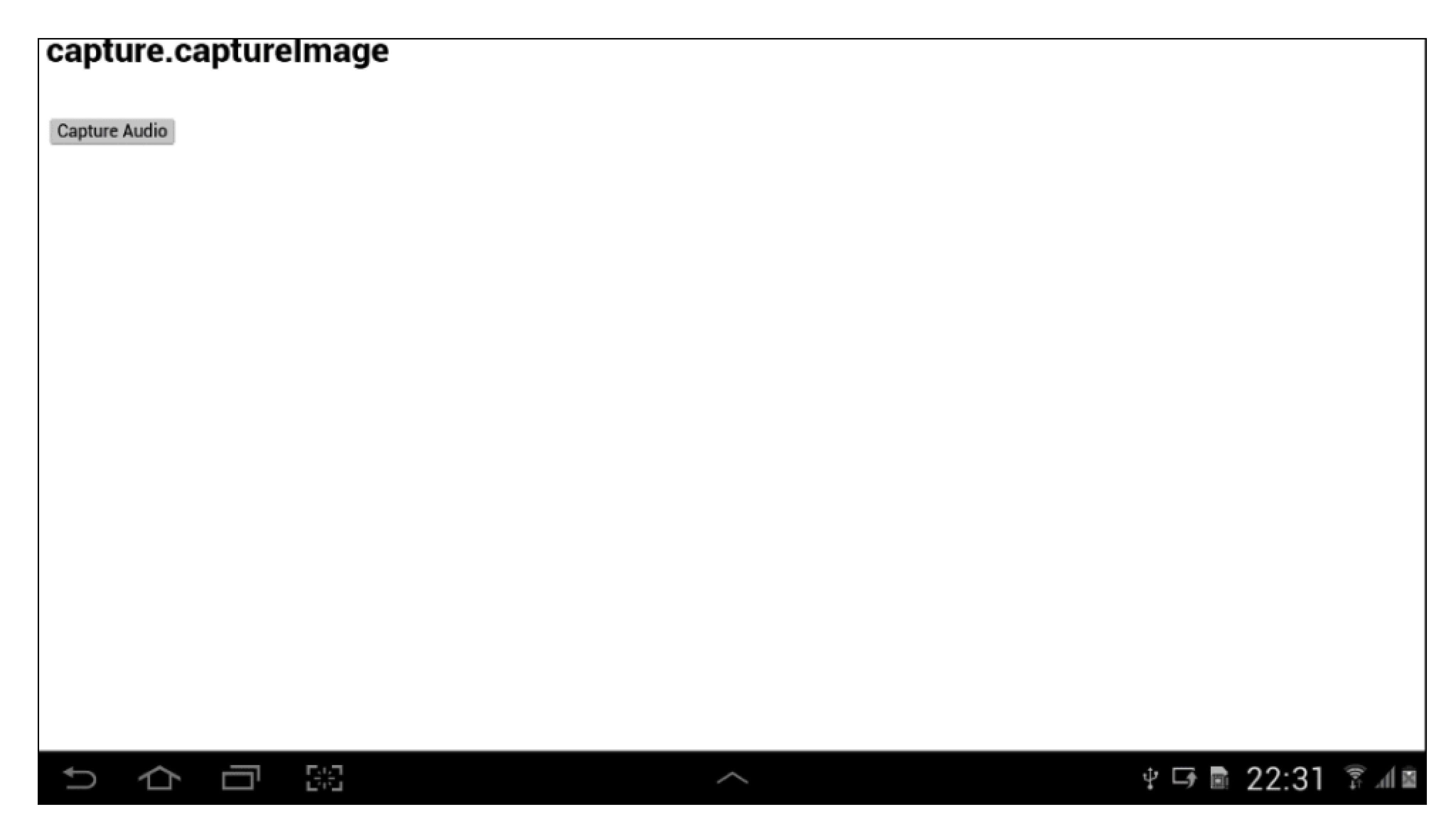


图 8.11　采集图像程序运行

拍照完成后，弹出文件地址和名称，会弹出两次，因为设置了拍摄两张照片，如图 8.12 所示。

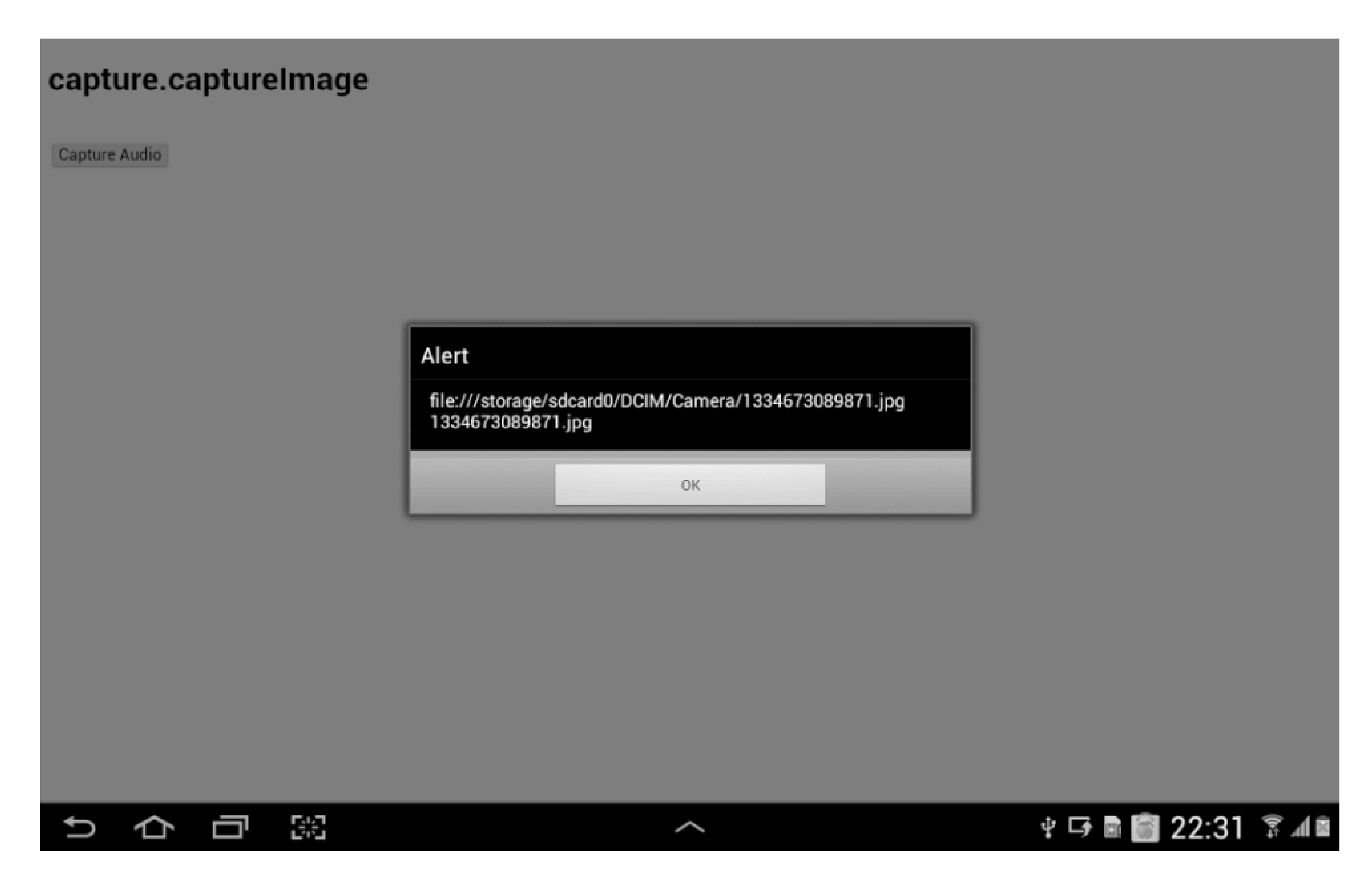


图 8.12　拍照完成

8.6　capture.captureVideo

capture.captureVideo 方法开启视频录制应用程序，返回采集到的视频剪辑文件信息。该方法通过设备的视频录制应用程序开始一个异步操作以采集视频录制。该操作允许设备用户在一个会话中同时采集多个视频录制。当用户退出视频录制应用程序，或系统到达 CaptureVideoOptions 的 limit 参数所定义的最大录制数时都会停止采集操作。如果没有设置 limit 参数的值，则使用其默认值 1，也就是说，当用户录制到一个视频剪辑后采集操作就会终止。当采集操作结束后，系统会调用 CaptureCB 回调函数，传递一个包含每个采集到的视频剪辑文件的 MediaFile 对象数组。如果用户在完成一个视频剪辑采集之前终止采集操作，系统会调用 CaptureErrorCB 回调函数，并传递一个包含 CaptureError.CAPTURE_NO_MEDIA_FILES 错误代码的 CaptureError 对象。下面给出一个例子代码：

```
<!DOCTYPE html>
<html>
  <head>
    <title>capture.captureAudio</title>
    <script type="text/javascript" charset="utf-8"
            src="js/cordova-2.6.0.js">
    </script>
  <script type="text/javascript" charset="utf-8">
    <span style="white-space:pre"></span>
  //captureVideo 方法成功执行后回调函数
    function captureSuccess(mediaFiles) {
```

```
        var i, len;
        for (i = 0, len = mediaFiles.length; i < len; i += 1) {
         //对应的逻辑内容
            }
    }

   //captureVideo 方法执行失败后回调函数
   function captureError(error) {
       var msg = "An error occurred during capture: " + error.code;
       navigator.notification.alert(msg, null, "oh");
   }

   function captureVideo() {
    navigator.device.capture.captureVideo(captureSuccess,captureError, {limit: 2});
    }
    </script>
  </head>
  <body>
   <h1>capture.captureImage</h1><br/>
    <button onclick="captureVideo();">Capture Video</button><br>
  </body>
</html>
```

8.7 Compass

Compass 对象获得该设备的当前朝向。罗盘是一个检测设备方向或朝向的传感器，使用度作为衡量单位，取值范围从 0°到 359.99°。这个对象有以下几个方法：

- compass.getCurrentHeading(compassSuccess, compassError, compassOptions)：获取罗盘的当前朝向。
- compass.watchHeading：在固定的时间间隔获取罗盘朝向的角度。
- compass.clearWatch：停止 watch ID 参数指向的罗盘监视。

下面给出一段例子代码：

```
<!DOCTYPE html>
<html>
  <head>
    <title>Compass Example</title>
    <script type="text/javascript" charset="utf-8"
    src="js/cordova-2.6.0.js"></script>
    <script type="text/javascript" charset="utf-8">
    var watchID = null;
    //等待 Cordova 加载
    document.addEventListener("deviceready", onDeviceReady, false);
    //Cordova 加载完毕
    function onDeviceReady() {
        navigator.compass.getCurrentHeading(function(heading){
```

```
            alert("Heading: " + heading.magneticHeading);
        }, function(compassError){
            alert("Compass Error: " + compassError.code);
        });
        startWatch();
    }

    //开始监听 Compass 罗盘
    function startWatch() {
        //每隔 3 秒获取一次 Compass 数据
        /**
          * options 选项
          * frequency: 多少毫秒获取一次罗盘朝向。(数字类型)(默认值: 100)
          * filter: 能够触发 watchHeadingFilter success 回调的罗盘改变度数(数字类型)
          * filter 在某些设备上不支持,具体参看文档
          */
        var options = { frequency: 3000 };
        watchID = navigator.compass.watchHeading(onSuccess, onError, options);
    }
    //停止对 Compass 的监听
    function stopWatch() {
        if (watchID) {
            navigator.compass.clearWatch(watchID);
            watchID = null;
        }
    }
    function onSuccess(heading) {
        var element = document.getElementById("heading");
        element.innerHTML = "Heading: " + heading.magneticHeading;
    }
    function onError(compassError) {
        alert("Compass error: " + compassError.code);
    }
    </script>
  </head>
  <body>
    <div id="heading">Waiting for heading...</div>
    <button onclick="startWatch();">Start Watching</button>
    <button onclick="stopWatch();">Stop Watching</button>
  </body>
</html>
```

8.8 Connection

Connection 对象就是去判断现在连接的网络,在程序中就可以使用 PhoneGap 的这个对象去判断当前的网络,然后做出对应的行为。这个对象只有一个 type 属性,可以取得这个 type 属性的值,然后再判断。type 属性的可选值,在代码中可以看到。下面给出一段例

子代码：

```
<!DOCTYPE html>
<html>
  <head>
    <title>navigator.connection.type Example</title>
    <script type="text/javascript" charset="utf-8"
    src="cordova-2.6.0.js"></script>
    <script type="text/javascript" charset="utf-8">
    document.addEventListener("deviceready", onDeviceReady, false);
    function onDeviceReady() {
        checkConnection();
    }
    function checkConnection() {
        var networkState = navigator.connection.type;
        var states = {};
        states[Connection.UNKNOWN]  = "Unknown connection";       //未知连接
        states[Connection.ETHERNET] = "Ethernet connection";      //以太网
        states[Connection.WIFI]     = "WiFi connection";          //wifi
        states[Connection.CELL_2G]  = "Cell 2G connection";       //2G
        states[Connection.CELL_3G]  = "Cell 3G connection";       //3G
        states[Connection.CELL_4G]  = "Cell 4G connection";       //4G
        states[Connection.CELL]     = "Cell generic connection";  //蜂窝网络
        states[Connection.NONE]     = "No network connection";
        alert("Connection type: " + states[networkState]);
    }
    </script>
  </head>
  <body>
    <p>弹出一个 alert 对话框显示当前的网络状态</p>
  </body>
</html>
```

8.9 Contacts

在手机中，程序可以访问手机的通讯录，这个时候需要使用 PhoneGap 提供的 Contacts 对象。Contacts 对象有以下几个方法。

- create 方法：创建联系人，返回一个 Contact 对象。
- find 方法：查询设备通讯录数据库，并返回包含指定字段的一个或多个 Contact 对象。find 方法的两个属性：contactFields、contactFindOptions。
  ✍contactFields——contacts.find 方法的必填参数，该参数定义了查找操作返回的 Contact 对象中应该包含哪些字段。
  ✍contactFindOptions——contacts.find 方法的可选参数，通过该参数从通讯录数据库中筛选联系人。
- clone：返回一个新的 Contact 对象，它是调用对象的深度拷贝，其 id 属性被设为 null。

• remove：从通讯录数据库中删除联系人。当删除不成功的时候，触发以 ContactError 对象为参数的错误处理回调函数。
• save：将一个新联系人存储到通讯录数据库，如果通讯录数据库中已经包含与其 ID 相同的记录，则更新该已有记录。

Contact 对象有以下的属性。

• id：全局唯一标识符(DOMString 类型)。
• displayname：联系人显示名称，适合向最终用户展示的联系人名称(DOMString 类型)。
• name：联系人姓名所有部分的对象(CantactName 类型)。
• nickname：昵称，对联系人的非正式称呼(DOMString 类型)。
• phoneNumbers：联系人所有联系电话的数组(ContactField[]类型)。
• emails：联系人所有 email 地址的数组(ContactField[]类型)。
• addrsses：联系人所有联系地址的数组。(ContactAddresses[]类型)
• ims：联系人所有 IM 地址的数组(ContactField[]类型)。
• organizations：联系人所属所有组织的数组(ContactOrganization[]类型)。
• birthday：联系人的生日(日期类型)。
• note：联系人的注释信息(DOMString 类型)。
• photos：联系人所有照片的数组(ContactField[]类型)。
• categories：联系人所属的所有用户自定义类别的数组(ContactField[]类型)。
• urls：与联系人相关网页的数组(ContactField[]类型)。

下面写一个例子代码：

```
<!DOCTYPE html>
<html>
  <head>
    <title>Contact Example</title>
    <script type="text/javascript" charset="utf-8"
    src="cordova-2.6.0.js"></script>
    <script type="text/javascript" charset="utf-8">
    document.addEventListener("deviceready", onDeviceReady, false);
    function onDeviceReady() {
        //create
        var contact = navigator.contacts.create();
        contact.displayName = "Plumber";
        contact.nickname = "Plumber";
        //specify both to support all devices
        var name = new ContactName();
        name.givenName = "Jane";
        name.familyName = "Doe";
        contact.name = name;
        //save
        contact.save(onSaveSuccess,onSaveError);
        //clone
        var clone = contact.clone();
```

```
            clone.name.givenName = "John";
            console.log("Original contact name = " + contact.name.givenName);
            console.log("Cloned contact name = " + clone.name.givenName);
            //remove
            contact.remove(onRemoveSuccess,onRemoveError);
        }
        function onSaveSuccess(contact) {
            alert("Save Success");
        }
        function onSaveError(contactError) {
            alert("save Error = " + contactError.code);
        }
        function onRemoveSuccess(contacts) {
            alert("Removal Success");
        }
        function onRemoveError(contactError) {
            alert("Remove Error = " + contactError.code);
        }
        </script>
      </head>
      <body>
        <h1>Example</h1>
        <p>Find Contacts</p>
      </body>
    </html>
```

8.10 Geolocation GPS 传感器

首先解释一下 Geolocation 这个单词，英语中是定位的意思，本书中翻译成了 GPS 传感器，主要是因为在官方文档中使用到了 GPS sensor，geolocation 对象提供了对设备 GPS 传感器的访问。geolocation 同样有以下三个方法。

- geolocation.getCurrentPosition：返回一个 Position 对象表示设备的当前位置。
- geolocation.watchPosition：定期查询设备的位置。
- geolocation.clearWatch：配合 watchPosition()使用，用于停止 watchPosition()查询。

下面写一个例子代码：

```
<!DOCTYPE html>
<html>
  <head>
    <title>Device Properties Example</title>
    <script type="text/javascript" charset="utf-8"
    src="cordova-2.6.0.js"></script>
    <script type="text/javascript" charset="utf-8">
    document.addEventListener("deviceready", onDeviceReady, false);
    function onDeviceReady() {
```

```
        navigator.geolocation.getCurrentPosition(onSuccess, onError);
    }
    function onSuccess(position) {
        var element = document.getElementById("geolocation");
        element.innerHTML = "Latitude 纬度:" + position.coords.latitude
          + "<br />" + "Longitude 经度: " + position.coords.longitude + "<br />" +
        "Altitude 位置相对于椭圆球面的高度:" + position.coords.altitude
         + "<br />" + "Accuracy 以米为单位的纬度和经度坐标的精度水平:"
         + position.coords.accuracy + "<br />" +
        "Altitude Accuracy 以米为单位的高度坐标的精度水平:" + position.coords.
        altitudeAccuracy +"<br />" +
        "Heading 运动的方向,通过相对正北做顺时针旋转的角度指定: " + position.coords.
        heading + "<br />" +
        "Speed 以米/秒为单位的设备当前地面速度: " + position.coords.speed
         + "<br />" + "Timestamp 以毫秒为单位的 coords 的创建时间戳: " +
        position.timestamp + "<br />";
    }

    function onError(error) {
        alert("code: " + error.code + "\n" + "message: " + error.message + "\n");
    }
    </script>
  </head>
  <body>
    <p id="geolocation">Finding geolocation...</p>
  </body>
</html>
```

8.11 InAppBrowser

InAppBrowser 对象是当用户调用 window.open 的时候,显示一个浏览器。在做项目的过程中客户需要在应用中打开另一个网站,这时候就可以直接使用这个对象,在应用程序中打开一个浏览器。window.open 方法会返回此对象。此对象有以下三个方法。

- addEventListener:增加事件监听器。
- removeEventListener:移除事件监听器。
- close:关闭对象。

这三个方法比较直观,下面介绍一下可以添加的事件。

- Loadstart:InAppBrowser 开始加载网页时候触发。
- Loadstop:InAppBrowser 网页加载完成触发。
- Loaderror:InAppBrowser 加载网页出错的时候触发。
- Exit:InAppBrowser 窗口关闭的时候触发。

下面看一个例子代码:

```
<!DOCTYPE html>
<html>
```

```
    <head>
      <title>InAppBrowser.removeEventListener Example</title>
      <script type="text/javascript" charset="utf-8"
      src="cordova-2.6.0.js"></script>
      <script type="text/javascript" charset="utf-8">
      document.addEventListener("deviceready", onDeviceReady, false);
      //Global InAppBrowser reference
      var iabRef = null;
      function iabLoadStart(event) {
          alert(event.type + " - " + event.url);
      }

      function iabLoadStop(event) {
          alert(event.type + " - " + event.url);
      }

      function iabLoadError(event) {
          alert(event.type + " - " + event.message);
      }

      function iabClose(event) {
           alert(event.type);
           iabRef.removeEventListener("loadstart", iabLoadStart);
           iabRef.removeEventListener("loadstop", iabLoadStop);
           iabRef.removeEventListener("loaderror", iabLoadError);
           iabRef.removeEventListener("exit", iabClose);
      }

      function onDeviceReady() {
           iabRef = window.open("http://www.baidu.com", "_blank", "location=yes");
           iabRef.addEventListener("loadstart", iabLoadStart);
           iabRef.addEventListener("loadstop", iabLoadStop);
           iabRef.removeEventListener("loaderror", iabLoadError);
           iabRef.addEventListener("exit", iabClose);
      }

      </script>
    </head>
    <body>
    </body>
</html>
```

8.12 Notification

Notification 对象主要是通知的功能，这个对象提供警告、确定、提示、鸣叫和振动 5 种通知，也就对应于下面 5 个方法。

- notification.alert：显示一个定制的警告或对话框，格式如下：

```
navigator.notification.alert(message,alertCallback,[title],[buttonName]);
```

✍message：对话框信息。(字符串类型)

✍alertCallback：当警告对话框被忽略时调用的回调函数。(函数类型)

✍title：对话框标题。(字符串类型)(可选项，默认值为“Alert”)

✍buttonName：按钮名称。(字符串类型)(可选项，默认值为“OK”)

- notification.confirm：显示一个可定制的确认对话框，格式如下：

```
navigator.notification.confirm(message,confirmCallback,[title],[buttonLabels])
```

✍message：对话框信息。(字符串类型)

✍confirmCallback：按下按钮后触发的回调函数，返回按下按钮的索引(1、2 或 3)。(函数类型)

✍title：对话框标题。(字符串类型)(可选项，默认值为“Confirm”)

✍buttonLabels：逗号分隔的按钮标签字符串。(字符串类型)(可选项，默认值为“OK、Cancel”)

- notification.prompt：显示一个定制的提示对话框，格式如下：

```
navigator.notification.prompt(message,promptCallback,[title],[buttonLabels])
```

✍message：对话框信息。(字符串类型)

✍promptCallback：按下按钮后触发的回调函数。(函数类型)

✍title：对话框标题。(字符串类型)(可选项，默认值为“Prompt”)

✍buttonLabels：显示按钮标签字符串的数组。(Array)(可选项，默认值为["OK","Cancel"])

- notification.beep：设备将发出蜂鸣声，格式如下：

```
navigator.notification.beep(times);
```

✍times：蜂鸣声的重复次数。(数字类型)

- notification.vibrate：使设备震动指定的时长，格式如下：

```
navigator.notification.vibrate(milliseconds);
```

✍time：以毫秒为单位的设备震动时长，1000 毫秒为 1 秒。(数字类型)

小　　结

在本章中，主要详细介绍了如何搭建 PhoneGap 开发环境，以及 Accelerometer、Camera、Capture、Compass、Connection、Contacts、Geolocation、InAppBrowser、Notification 等特性，通过本章的学习，读者可以利用 PhoneGap 去开发跨平台的移动应用程序。

教 学 资 源 支 持

敬爱的教师：

感谢您一直以来对清华版计算机教材的支持和爱护。为了配合本课程的教学需要，本教材配有配套的电子教案（素材），有需求的教师请到清华大学出版社主页（http://www.tup.com.cn）上查询和下载，也可以拨打电话或发送电子邮件咨询。

如果您在使用本教材的过程中遇到了什么问题，或者有相关教材出版计划，也请您发邮件告诉我们，以便我们更好地为您服务。

我们的联系方式：

地　　址：北京海淀区双清路学研大厦 A 座 707

邮　　编：100084

电　　话：010－62770175－4604

课件下载：http://www.tup.com.cn

电子邮件：weijj@tup.tsinghua.edu.cn

教师交流 QQ 群：136490705

教师服务微信：itbook8

教师服务 QQ：883604

（申请加入时，请写明您的学校名称和姓名）

用微信扫一扫右边的二维码，即可关注计算机教材公众号。



扫一扫

课件下载、样书申请

教材推荐、技术交流